

2

DTIC FILE COPY

# NAVAL POSTGRADUATE SCHOOL Monterey, California

AD-A205 071



## THESIS

DTIC  
ELECTE  
MAR 1 0 1989  
S QH D

**THE APPLICATION OF BRIAN'S METHOD  
TO THE SOLUTION OF  
TRANSIENT HEAT CONDUCTION PROBLEMS  
IN CYLINDRICAL GEOMETRIES**

by

Karl R. Heinz

December 1988

Thesis Advisor:

Yogendra Joshi

Approved for public release; distribution is unlimited.

89 3 09 032

Unclassified

Security Classification of this page

## REPORT DOCUMENTATION PAGE

1a Report Security Classification <b>Unclassified</b>		1b Restrictive Markings	
2a Security Classification Authority		3 Distribution Availability of Report	
2b Declassification/Downgrading Schedule		Approved for public release; distribution is unlimited.	
4 Performing Organization Report Number(s)		5 Monitoring Organization Report Number(s)	
6a Name of Performing Organization	6b Office Symbol (If Applicable) 69	7a Name of Monitoring Organization	
Naval Postgraduate School		Naval Postgraduate School	
6c Address (city, state, and ZIP code)		7b Address (city, state, and ZIP code)	
Monterey, CA 93943-5000		Monterey, CA 93943-5000	
8a Name of Funding/Sponsoring Organization	8b Office Symbol (If Applicable)	9 Procurement Instrument Identification Number	
8c Address (city, state, and ZIP code)		10 Source of Funding Numbers	
		Program Element Number Project No Task No Work Unit Accession No	
11 Title (Include Security Classification) The Application of Brian's Method to the Solution of Transient Heat Conduction Problems in Cylindrical Geometries			
12 Personal Author(s) Karl R. Heinz			
13a Type of Report	13b Time Covered	14 Date of Report (year, month, day)	15 Page Count
Master's Thesis	From To	December 1988	182
16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
17 Cosati Codes		18 Subject Terms (continue on reverse if necessary and identify by block number)	
Field	Group	Subgroup	
		Brian's Method, finite difference analysis, transient heat conduction, cylinder, cylindrical geometry, theses.	
19 Abstract (continue on reverse if necessary and identify by block number)			
A FORTRAN 77 computer code employing an adaptation of the finite differencing algorithm proposed by Brian was developed for the solution of transient heat conduction problems in cylindrical geometries. Validation of code was accomplished by comparison with an analytic solution derived for a model with symmetric, linear boundary conditions. Accuracy of results for asymmetric and non-linear boundary conditions was determined by comparison with a similarly validated code employing the explicit method. Code effectiveness was then demonstrated by conducting a transient temperature analysis for a simulated earth-orbiting satellite. Brian's method demonstrated unconditional stability with associated significant reductions in execution time compared to the explicit method. The effects of discretization error on the accuracy of results require further investigation.			
20 Distribution/Availability of Abstract		21 Abstract Security Classification	
<input checked="" type="checkbox"/> unclassified/unlimited <input type="checkbox"/> same as report <input type="checkbox"/> DTIC users		Unclassified	
22a Name of Responsible Individual		22b Telephone (Include Area code)	22c Office Symbol
Professor Yogendra Joshi		(408) 646-3400	69Ji

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted

security classification of this page

All other editions are obsolete

Unclassified

Approved for public release; distribution is unlimited.

**The Application of Brian's Method to the Solution of Transient  
Heat Conduction Problems in Cylindrical Geometries**

by

Karl R. Heinz  
Lieutenant Commander, United States Navy  
B.S., University of Kansas, 1976

Submitted in partial fulfillment of the  
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

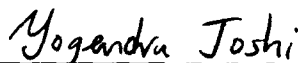
from the

NAVAL POSTGRADUATE SCHOOL  
December 1988

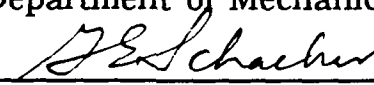
Author:

  
Karl R. Heinz

Approved by:

  
Yogendra Joshi, Thesis Advisor

  
Anthony J. Healey, Chairman,  
Department of Mechanical Engineering

  
Gordon E. Schacher, Dean of  
Science and Engineering

## ABSTRACT

A FORTRAN 77 computer code employing an adaptation of the finite differencing algorithm proposed by Brian was developed for the solution of transient heat conduction problems in cylindrical geometries. Validation of code was accomplished by comparison with an analytic solution derived for a model with symmetric, linear boundary conditions. Accuracy of results for asymmetric and non-linear boundary conditions was determined by comparison with a similarly validated code employing the explicit method. Code effectiveness was then demonstrated by conducting a transient temperature analysis for a simulated earth-orbiting satellite. Brian's method demonstrated unconditional stability with associated significant reductions in execution time compared to the explicit method. The effects of discretization error on the accuracy of results require further investigation.



**Accession For**

NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	

DATE \_\_\_\_\_ BY \_\_\_\_\_

FILE NO. \_\_\_\_\_

REPORT NUMBER \_\_\_\_\_

A-1

## TABLE OF CONTENTS

<b>I</b>	<b>INTRODUCTION.....</b>	<b>1</b>
A	BACKGROUND.....	1
B	PROBLEM.....	1
C	REQUIREMENT.....	3
D	OBJECTIVES.....	3
<b>II.</b>	<b>THEORY.....</b>	<b>4</b>
A	FUNDAMENTALS.....	4
1.	Material Properties .....	4
a	Heat Capacity .....	4
b.	Thermal Conductivity.....	5
c.	Thermal Diffusivity .....	5
d.	Convection Heat Transfer Coefficient.....	5
e.	Properties Affecting Radiation.....	6
2.	Heat Transfer Rate .....	7
3.	Rate Equations .....	8
a	Conduction Equation .....	8
b.	Convection Equation.....	8
c.	Radiation Equation .....	8
4.	Heat Balance—The First Law of Thermodynamics.....	9

B.	FINITE DIFFERENCE APPROXIMATIONS.....	10
1.	General Approach .....	10
a.	Geometry.....	12
b.	Subscripting Convention.....	12
c.	Superscripting Convention .....	13
d.	Example.....	14
2.	The Explicit Method .....	16
3.	Brian's Method.....	17
III.	DERIVATION OF EQUATIONS.....	19
A.	MODEL FOR CODE DEVELOPMENT.....	19
1.	General.....	19
2.	Problem Description.....	19
B.	EXPLICIT METHOD EQUATIONS .....	20
C.	BRIAN'S METHOD.....	27
1.	Step One (z-Direction).....	27
2.	Step Two ( $\phi$ -Direction).....	29
3.	Step Three (r-Direction) .....	30
4.	Step Four (New Temperatures) .....	32
IV.	CODE DEVELOPMENT .....	33
A.	ANALYTIC SOLUTION .....	33
B.	EXPLICIT METHOD .....	33
1.	Requirements.....	33
2.	General Procedure.....	33

a	Timestep Restriction .....	33
b	Indexing.....	34
C	BRIAN'S METHOD.....	35
1.	Requirements.....	35
a	Timestep .....	35
b	Indexing.....	35
D	EXECUTION TIMEKEEPING .....	40
V.	VALIDATION .....	41
A	INITIAL CODE .....	41
1.	General Procedure.....	41
2.	Time Incrementation .....	41
B	APPLICATION CODE .....	42
VI.	RESULTS.....	44
A	DEFINITIONS.....	44
1.	Convergence.....	44
2.	Steady-State .....	44
B	VALIDATION .....	44
1.	Analytic Solution.....	44
2.	Explicit Method.....	44
3.	Brian's Method.....	45
C	FOLLOW-ON TESTING .....	45
1.	Symmetric Boundary Conditions .....	48
2.	Asymmetric Boundary Conditions .....	48

D.	APPLICATION EXAMPLE.....	49
<b>VII.</b>	<b>CONCLUSIONS AND RECOMMENDATIONS.....</b>	<b>52</b>
A.	CONCLUSIONS.....	52
1.	Adaptability.....	52
2.	Relative Accuracy.....	52
3.	Relative Execution Speed.....	52
4.	Relative Benefits.....	52
B.	RECOMMENDATIONS.....	53
1.	Adaptation of Brian's Method FORTRAN Code to C Programming Language.....	53
a	Background.....	53
b.	Recommendation.....	53
2.	Incorporation of the Analysis of Directional and Time-Dependent Properties in Brian's Method.....	54
a	Background.....	54
b.	Recommendation.....	54
APPENDIX A	NODE EQUATIONS FOR EXPLICIT METHOD.....	55
APPENDIX B	NODE EQUATIONS FOR BRIAN'S METHOD.....	61
APPENDIX C	VALIDATION MODEL.....	78
APPENDIX D	EXPLICIT METHOD STABILITY REQUIREMENTS....	85
APPENDIX E	EXPLICIT METHOD CODE.....	87
APPENDIX F	BRIAN'S METHOD CODE.....	109
APPENDIX G	APPLICATION MODEL CHARACTERISTICS.....	169
	LIST OF REFERENCES.....	172
	INITIAL DISTRIBUTION LIST.....	173



## **ACKNOWLEDGMENTS**

The author wishes to extend his sincerest thanks to Assistant Professor Yogendra Joshi who provided the insight, technical expertise, and patience required to see this project through to fruition; to the faculty of the Mechanical Engineering Department of the Naval Postgraduate School, whose skillful instruction presented the necessary tools for the task; to Mr. Alvin Lau, for his perseverance in the provision of graphic arts; and to Maria, Ryan, and Karan, without whose patience and loving support this task would not have been possible.

## I. INTRODUCTION

### A. BACKGROUND

Analysis of transient heat-conduction problems necessarily involves solution of the heat diffusion equation. In the absence of internal generation of energy, this is represented by

$$\nabla \cdot (k \nabla T) = \rho C_p \frac{\partial T}{\partial x} \quad (1.1)$$

Although analytic solutions may be obtained to certain simple transient problems [Ref. 1:p. 212], many of the practical problems encountered by engineers involve one or a combination of nonlinearities, complex geometries, complex boundary conditions, or systems of coupled partial differential equations of which any one may necessitate the use of numerical methods. Although Monte-Carlo (probability sampling) and finite-element methods have been applied to the solution of heat-conduction problems, the approximation of partial derivatives by finite differences presents a straightforward and popular approach [Ref. 2:p. 471]. Two fundamental finite-difference techniques are the implicit and explicit methods and their derivatives.

### B. PROBLEM

The analysis of transient heat-conduction problems in three dimensions is of great importance to many fields of science and engineering, but solutions are often quite costly in terms of computational

effort and time consumed in their solution. Numerical solutions to these problems frequently tax the computational and storage capacities of available computers [Ref. 3:p. 367].

The *explicit method* offers the benefits of relatively low storage requirements and simplicity in formulation, coding, and execution. However, it is only conditionally stable and is subject to a limitation on the maximum timestep which may be employed. If the maximum timestep is exceeded, the solution experiences numerically induced oscillations which may cause the solution to diverge from the correct result [Ref. 1:p. 214]. This timestep restriction causes the explicit method to be inefficient in terms of computational effort for the analysis of lengthy transient periods.

The *implicit method* offers the advantage of being unconditionally stable and therefore exempt from timestep restrictions. However, this method calculates node temperatures by the simultaneous solution of the nodal heat balance equations. Thus, savings gained from the lack of a timestep restriction are often quickly offset by the increased computational effort associated with the required matrix inversions.

The *implicit alternating direction method* (ADI) is an unconditionally stable extension of the implicit method which reduces the number of required computations by transforming the problem into tridiagonal form. However, a direct extension of this method into three dimensions becomes only conditionally stable, is restricted to a maximum timestep in a fashion similar to the explicit method, and demonstrates similar shortcomings [Ref. 4:p. 453].

### **C. REQUIREMENT**

A finite difference technique which is unconditionally stable in three dimensions and unencumbered by the requirement to manipulate large matrices is required for the solution of transient heat-conduction problems in three dimensions. The algorithm proposed by Brian [Ref. 3] offers such a technique.

### **D. OBJECTIVES**

1. Adapt the algorithm proposed by Brian to the solution of three-dimensional transient heat-conduction problems in cylindrical geometries.
2. Evaluate the relative accuracy of the results obtained using Brian's Method by comparing them against those obtained using the explicit method.
3. Evaluate the speed of execution of Brian's Method relative to that of the explicit method in terms of central processing unit (CPU) seconds.
4. Illustrate the relative benefits derived from the Brian's Method by applying it to the solution of an application problem involving long-period transient behavior.

## II. THEORY

### A. FUNDAMENTALS

#### 1. Material Properties

The capacity for energy storage and rate of energy transfer within a medium are functions of the material's thermophysical properties. Isotropic materials, such as metals and alloys, are uniform in nature and their properties generally demonstrate only negligible directional and temperature dependence. However, layered or stratified materials, such as glass-reinforced plastic (GRP) or laminates, may demonstrate strong dependence on both direction and temperature. While directional dependence of properties is generally linear in nature, temperature dependence is strongly nonlinear and will not be addressed.

##### a Heat Capacity

The rate of change of energy storage per unit volume within a solid, isotropic medium is determined by the product of its density ( $\text{Kg/m}^3$ ), specific heat ( $\text{KJ/KgK}$ ), and the time-rate of change of its temperature:

$$\dot{E}_s = \rho C_p \frac{\partial T}{\partial t} \quad (2.1)$$

Both density and specific heat may be temperature and/or directionally dependent properties.

**b. Thermal Conductivity**

The thermal conductivity (W/mK) is a transport property (i.e., one affecting the movement of energy) of the material and is defined in the x-coordinate direction as

$$k_x = - \frac{q''_x}{\left( \frac{\partial T}{\partial x} \right)} \quad (2.2)$$

Isotropic materials generally demonstrate relatively constant thermal conductivity over broad temperature ranges, whereas metallic laminates demonstrate a strong directional dependence within the plane of each different material. Composite materials show both strong directional and temperature dependence. Temperature-dependent material properties will not be addressed.

**c. Thermal Diffusivity**

The thermal diffusivity (m<sup>2</sup>/s) is a grouping of the material density, specific heat, and thermal conductivity and represents the controlling transport property for transient heat conduction [Ref. 1:p. 41]

$$\alpha = \frac{k}{\rho C_p} \quad (2.3)$$

**d. Convection Heat Transfer Coefficient**

The convection heat transfer coefficient (W/m<sup>2</sup>K) encompasses all of the effects that influence the convection mode and is dependent upon boundary conditions, surface geometry, the nature

of the fluid motion, and a number of other fluid thermodynamic and transport properties [Ref. 1:p. 8].

**e. Properties Affecting Radiation**

Various surface properties affect the radiative characteristics of a solid material. Directionally integrated or hemispherical values are often adequate although these properties are still strongly dependent upon the wavelength of the radiant energy. Where appropriate, consideration of the wavelength dependence will be indicated by a subscripted Greek letter lamda ( $\lambda$ ).

**(1) Emissivity.** The *emissivity* ( $\epsilon$ ) is a surface property representing the ratio of total energy emitted by an actual surface and an identical black surface at the same temperature. The emissivity may be altered by one or a combination of mechanical processes (e.g., sandblasting or metal-spraying) or by application of various films and/or coatings, either by design or through service conditions.

**(2) Surface Properties.** *Absorptivity* ( $\alpha$ ) and *reflectivity* ( $\rho$ ) are surface properties representing the fractions of total incident energy which are respectively absorbed and reflected by the surface. For opaque surfaces, the sum of the absorptivity and the reflectivity equals one

$$\alpha + \rho = 1 \quad (2.4)$$

In addition to absorption and reflection, translucent materials allow a fraction of incident radiation to be transmitted through the material.

Consequently, an additional property, *transmissivity* ( $\tau$ ), must be incorporated into Equation 2.4, yielding

$$\alpha + \rho + \tau = 1 \quad (2.5)$$

**(3) Radiation Heat Transfer Coefficient.** The *radiation heat transfer coefficient* ( $\text{W/m}^2\text{K}^4$ ) is a collection of terms resulting from the linearization of the radiation heat transfer calculation from a gray, diffuse surface in extensive surroundings and is represented as

$$h_r = \epsilon \sigma (T_s + T_{sur})(T_s^2 + T_{sur}^2) \quad (2.6)$$

where  $\sigma$  is the *Stefan-Boltzman Constant* ( $5.67 \times 10^{-8} \text{ W/m}^2\text{K}$ ) and  $T_s$  and  $T_{sur}$  are the absolute temperatures of the surface and the surroundings, respectively. As is apparent from Equation 2.6, this term is strongly temperature dependent and non-linear in nature.

## 2. Heat Transfer Rate

The *heat flux* ( $\text{W/m}^2$ ) is rate of heat transfer per unit surface area and may be represented by

$$q'' = \frac{\dot{E}}{A} \quad (2.7)$$

The *heat transfer rate* ( $\text{W/m}^2\text{s}$ ) is thus the product of the heat flux and the surface area normal to the direction of heat flow [Ref. 1:p. 4].



### 3. Rate Equations

The three principle modes of heat transfer are *conduction*, *convection*, and *radiation*. The rate of heat transfer attributed to each mode is described by the appropriate *rate equation*.

#### a Conduction Equation

The governing equation for conduction is *Fourier's Law*, given for the x-direction by

$$q''_x = -k_x \frac{\partial T}{\partial x} \quad (2.8)$$

#### b. Convection Equation

The relation governing convection is known as *Newton's Law of Cooling* and is given by

$$q''_c = h_c (T_s - T_{sur}) \quad (2.9)$$

where  $T_s$  is the surface temperature and  $T_\infty$  is the temperature of the surrounding fluid.

#### c. Radiation Equation

The *maximum* flux at which radiation may be emitted by a surface to an infinite, black surrounding is determined by the *Stefan-Boltzman Law*

$$q''_r = h_r (T_s - T_{sur}) \quad (2.10)$$

where  $T_s$  is the surface temperature and  $T_{sur}$  is the temperature of the surrounding enclosure [Ref. 1:p. 9]. In reality, objects exchange

energy with surrounding enclosure boundaries in accordance with the relation

$$q'' = \sum_i F_i A_i h_r (T_{s_i} - T_{sur_i}) \quad (2.11)$$

where  $F$  is the *view factor* representing the fraction of the radiation emitted by the surface of interest that is intercepted by the  $i^{\text{th}}$  surface of the confining enclosure. Implicit in Equation 2.11 is the assumption that all surfaces of the surrounding enclosures are black.

#### 4. Heat Balance—The First Law of Thermodynamics

The temperature at a given point may be determined from establishment of an arbitrary control volume around the point (Figure 1) followed by application of the first law of thermodynamics (the *law of conservation of energy*). Excluding heat which is generated within the confines of the control volume (internal generation), this law may be simply stated as

The rate at which thermal and mechanical energy enters a control volume minus the rate at which this energy leaves the control volume must equal the rate at which this energy is stored in the control volume. [Ref. 1:p. 12]

With the addition of any heat generated internal to the control volume, the first law may be represented in the form of the *heat balance equation*:

$$\dot{E}_s = \dot{E}_{in} + \dot{E}_{gen} - \dot{E}_{out} \quad (2.12)$$

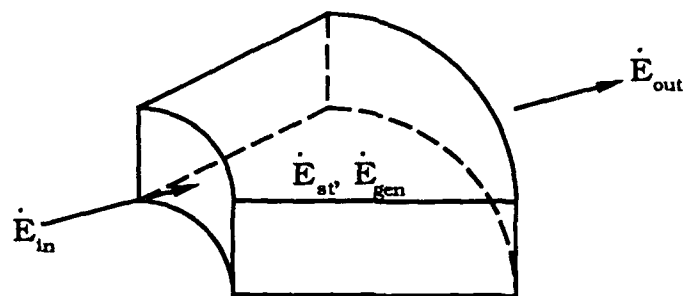


Figure 1. Control Volume

where

$\dot{E}_s$  = Rate of energy storage

$\dot{E}_{in}$  = Rate of energy gain

$\dot{E}_{out}$  = Rate of energy loss

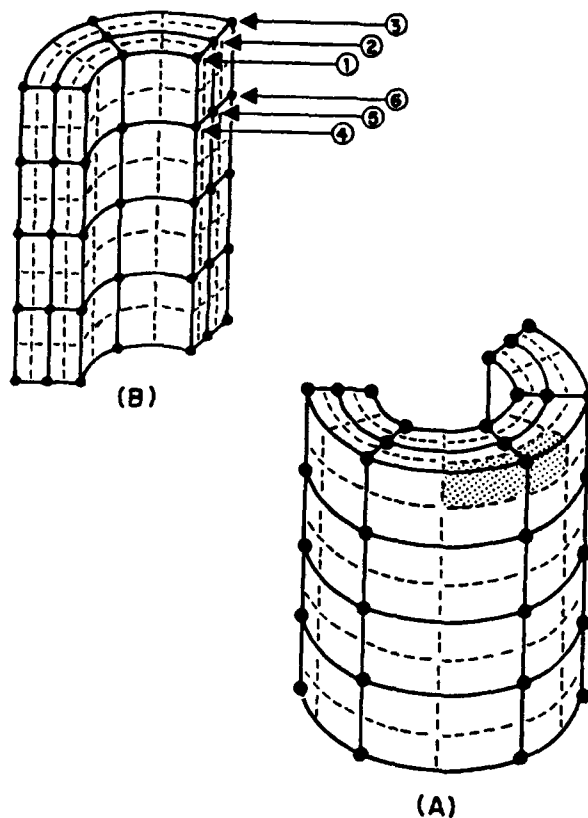
$\dot{E}_{gen}$  = Rate of energy generation.

## B. FINITE DIFFERENCE APPROXIMATIONS

### 1. General Approach

The numerical solution of transient, heat-conduction problems by digital computer requires the transformation of the heat conduction equation into a form which allows differentiation to be approximated by finite differences [Ref. 2:p. 472]. This may be accomplished by dividing the problem into a number of discrete points, or nodes, each surrounded by an associated control volume (Figure 2(A)). Conducting an energy balance on each control volume yields a system of linear algebraic equations which describes the temperature field. In the absence of round-off error, the solutions obtained from finite

difference approximations tend to the analytic solution as the step size and grid spacings tend to zero [Ref. 4:p. 449].



- Type 1—End-layer, inner-surface
- Type 2—End-layer, mid-radius
- Type 3—End-layer, outer-surface
- Type 4—Mid-layer, inner-surface
- Type 5—Mid-layer, mid-radius
- Type 6—Mid-layer, outer-surface

Figure 2. (A) **Reference Grid**; (B) **Node Types**

### **a. Geometry**

Figure 2(B) depicts a right-circular cylinder to which has been applied a polar-cylindrical coordinate system and an arbitrary grid which divides the cylinder into a number of nodes, each surrounded by an associated control volume. Nodes may be characterized as belonging to one of six types and are located at the grid intersection points. Each node may be associated with a specified layer, ring, and ray.

(1) **Layers.** Layers lie parallel to the x-y plane and are numbered sequentially beginning in the x-y plane and proceeding in the direction of the positive z-axis. Layers are spaced a distance  $\Delta z$  apart. Note that the end layers are in the plane containing the ends of the cylinders.

(2) **Rings.** Each layer is divided into an arbitrary number of concentric rings spaced a distance  $\Delta r$  apart. The node at the three o'clock position when viewed from above has been arbitrarily designated as node one. Other nodes are numbered sequentially in a counter-clockwise direction from node one.

(3) **Rays.** Rays emanate outward from the origin in the plane of the layers. Nodes lie along the rays at intervals of  $\Delta r$  and are numbered sequentially from the inner surface outward.

### **b. Subscripting Convention**

To simplify notation in the development of equations, the designation of nodal grid coordinates will follow the convention contained in Table 1.

TABLE 1  
NODAL SUBSCRIPT CONVENTION

Subscript	Location	Grid Coordinates*
N	Northern (upper) neighbor	$[r, \phi, (z+1)]$
S	Southern (lower) neighbor	$[r, \phi, (z-1)]$
E	Eastern (right) neighbor	$[r, (\phi+1), z]$
W	Western (left) neighbor	$[r, (\phi-1), z]$
I	Inner neighbor	$[(r-1), \phi, z]$
O	Outer neighbor	$[(r+1), \phi, z]$
e	Edge	(varies)

\* All grid coordinates are relative to the node of interest when viewed from the inside of the cylinder.

In view of the inverse symmetric relationship existing between equivalent nodes at the upper and lower surfaces, the subscript *e* will be used to designate temperatures, fluxes, and properties associated with the end layers, thus reducing the number of required equations by half. The reader must substitute the appropriate subscript (i.e., *N* or *S*) depending upon the location or application of the equation. This issue will be further elaborated upon in the equation development example.

**c. Superscripting Convention**

Superscripts, as contained in Table 2, will be employed to indicate at which timestep a value is considered.

TABLE 2  
NODAL SUPERScript CONVENTION

Superscript	Timestep	Time
n	Current	t
n+1	Next	t + $\Delta t$
n+1/2	Intermediate	t + $\Delta t/2$

**d. Example**

Consider the one-dimensional heat conduction problem represented in Figure 3 wherein  $T_E = T_W$ , heat is being conducted in the radial direction only, and the control volume is of unit thickness.

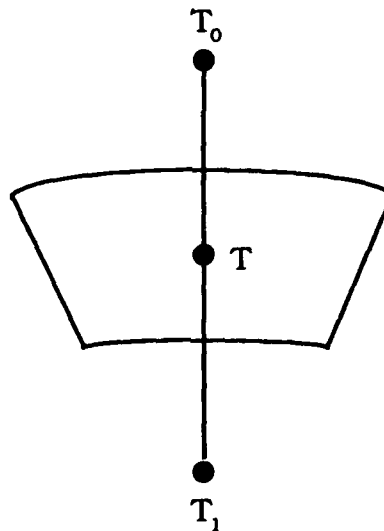


Figure 3. One-Dimensional Heat Conduction Problem

The appropriate heat balance equation is

$$\dot{E}_s = \dot{E}_i - \dot{E}_o \quad (2.13)$$

The time-dependent storage term ( $\dot{E}_s$ ) may be represented by its finite difference approximation

$$\dot{E}_s = \rho C_p V \frac{(T^{n+1} - T^n)}{\Delta t} \quad (2.14)$$

where  $\Delta t$  is the step size,  $T^n$  is the node temperature at the *old* (n) timestep,  $T^{n+1}$  is the node temperature at the *new* (n+1) timestep, and the volume is approximated as

$$V \cong \frac{r \Delta r \Delta \phi}{2} \quad (2.15)$$

The heat rate in the radial direction at the inner boundary ( $\dot{E}_i$ ) of the control volume may be approximated as

$$\dot{E}_i = -k_r A_i \frac{(T_i^n - T^{n+1})}{\Delta r} \quad (2.16)$$

where

$$A_i = (r - \frac{\Delta r}{2}) \Delta \phi \quad (2.17)$$

is the area of the control volume in the direction of heat flow. Similarly, the heat rate at the outer boundary may be approximated as

$$\dot{E}_o = -k_r A_o \frac{(T_o^n - T^{n+1})}{\Delta r} \quad (2.18)$$

where

$$A_o = (r + \frac{\Delta r}{2}) \Delta r \Delta \phi \quad (2.19)$$



Substituting these into Equation 2.13 yields the finite difference form of the heat equation

$$\frac{\rho C_p r \Delta r \Delta \phi}{2} \frac{(T^{n+1} - T^n)}{\Delta t} = \frac{k_r (r - \frac{\Delta r}{2}) \Delta \phi}{\Delta r} (T_i^n - T^n) \quad (2.20) \\ + \frac{k_r (r + \Delta r) \Delta \phi}{\Delta r} (T_o^n - T^n)$$

The steps leading up to and including the formation of Equation 2.20 are common to both the explicit and Brian's methods.

Similar equations may be obtained for other nodes and boundary conditions.

## 2. The Explicit Method

The explicit method is a direct application of the general approach to finite difference analysis detailed in the preceding section. Equations describing the temperature at the new time level are formed for each node by conducting a heat balance on the associated control volume. Temperatures at the new time level may then be approximated explicitly by substituting temperatures at the old time level into the appropriate node equation. The explicit form of the one-dimensional heat conduction problem of the preceding section may be obtained from Equation 2.20 by dividing by the lead coefficient on the left-hand side, substituting for the grid Fourier number, and solving for the new temperature, yielding

$$T^{n+1} = T^n + Fo \left[ \frac{(r - \frac{\Delta r}{2})}{\Delta r} (T_i^n - T^n) + \frac{(r + \frac{\Delta r}{2})}{\Delta r} (T_o^n - T^n) \right] \quad (2.21)$$

where

$$Fo = \frac{\alpha \Delta t}{\Delta r^2} \quad (2.22)$$

Defining the constants

$$c_1 = \frac{(r - \frac{\Delta r}{2})}{\Delta r} \quad (2.23)$$

and

$$c_2 = \frac{(r + \frac{\Delta r}{2})}{\Delta r} \quad (2.24)$$

and grouping the  $T^n$  terms outside of the the parentheses yields

$$T^{n+1} = Fo(c_1 T_I^n + c_2 T_O^n) + [1 - Fo(c_1 + c_2)] T^n \quad (2.25)$$

from which the stability criterion on the timestep may be determined from the relation

$$[1 - Fo(c_1 + c_2)] \geq 0 \quad (2.26)$$

[Ref. 1:p. 215]

### 3. Brian's Method

In Brian's Method, the node types are the same as depicted in Figure 2. However, the temperature at the new time level is formed from a combination of values taken from the old time level and three

directionally dependent temperature arrays formed at an intermediate time

$$t' = t + \frac{\Delta t}{2} \quad (2.27)$$

Although each of these arrays requires a full complement of supporting node equations, the equations are quite similar and enjoy a common derivation with the explicit method for the initial stages of their development.

Each intermediate array is associated with one arbitrarily assigned coordinate direction and has its temperatures denoted by the superscripting of one, two, or three asterisks. *Calculation precedence must follow the order of star level, two-star level, three-star level, and new temperature arrays!*

Within each intermediate calculation process, temperatures are determined by solution of a tridiagonal matrix of coefficients of nodes lying along the specified coordinate axis. For example, equations for the r-direction nodes at the star level in a one-dimensional heat conduction problem are of the form

$$AT_i^* + BT^* + CT_o^* = D(T^n) \quad (2.23)$$

Although Brian's method is conceptually no more difficult than the explicit method, it involves considerably more algebra and is best illustrated in the form of the code development model contained in the following chapter.

### III. DERIVATION OF EQUATIONS

#### A. MODEL FOR CODE DEVELOPMENT

##### 1. General

In this chapter, detailed descriptions of the explicit and Brian's methods will be presented in the form of the model employed for the development of code.

##### 2. Problem Description

Selection of a model with simplicity and symmetry of geometry and boundary conditions was made to facilitate error checking and the comparison of results during code development. The preliminary model consisted of a right-circular cylinder of uniform properties with adiabatic ends and specified initial temperature distribution (initial condition) and constant inner-surface temperature of zero degrees Celsius. An arbitrary grid consisting of five layers in the z-direction, each with three and four nodes respectively in the radial and phi-directions, was applied to the cylinder in a fashion similar to that depicted in Figure 4. At time  $t = 0$ , a uniform, constant heat flux was applied to the outer surface. Subsequent modifications to the model added radiative and convective boundary conditions at all surfaces as well as directionally-dependent thermal conductivities. Temperatures are in terms of the rise above the initial temperature (*excess temperature*) [Ref. 1:p. 100].

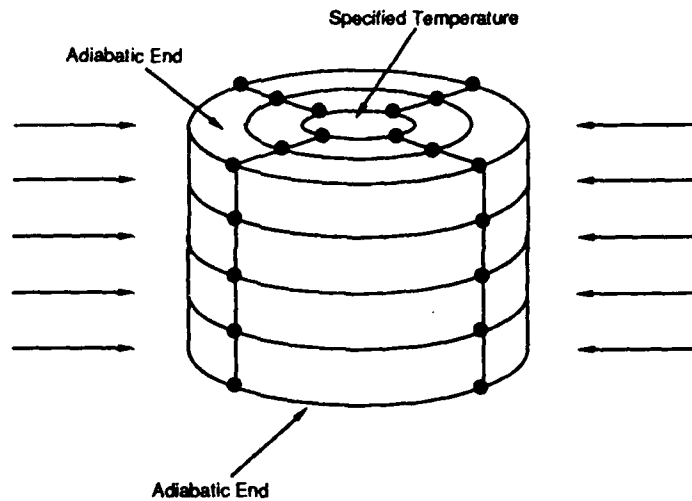
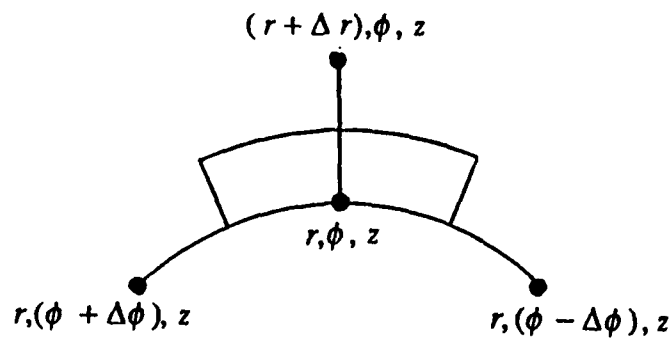


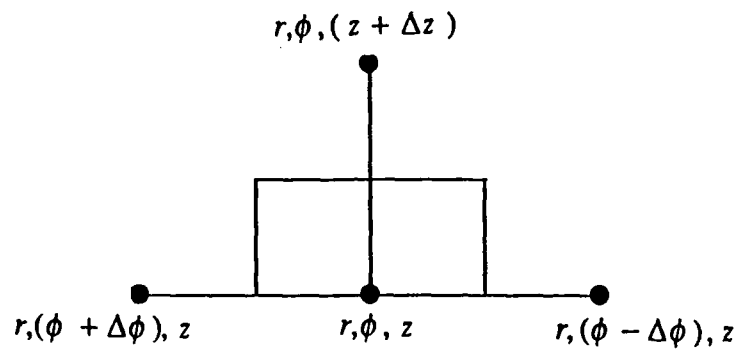
Figure 4. **Model for Code Development**

## **B EXPLICIT METHOD EQUATIONS**

The process for development of equations using the explicit method is similar for all node types. For illustrative purposes, we will consider a Type 1 node located at the inner edge of the bottom surface of the cylinder as depicted in Figure 5. Complete equations for all node types are included as Appendix A to this report. Directionally dependent thermal conductivities and radiative and convective boundary conditions are included in this development. Heat flow into and out of the associated control volume is represented in Figure 6.



A. Plan View



B. Frontal View

Figure 5. **Bottom Layer, Inner Surface Node Configuration**

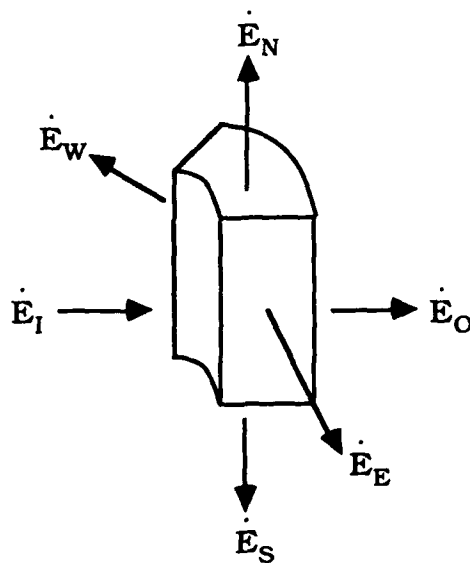


Figure 6. Heat Balance

The heat balance equation is

$$\dot{E}_s = \dot{E}_I - \dot{E}_N - \dot{E}_S - \dot{E}_E - \dot{E}_W - \dot{E}_O \quad (3.1)$$

Application of appropriate finite difference approximations results in

$$\begin{aligned} \rho C_p V \frac{(T^{n+1} - T^n)}{\Delta t} = & k_z A_z \frac{(T_N^n - T^n)}{\Delta z} + k_\phi A_\phi \frac{(T_E^n + T_W^n - 2T^n)}{r \Delta \phi} \\ & + k_r A_r \frac{(T_O^n - T^n)}{\Delta r} - \dot{E}_S + \dot{E}_I \end{aligned} \quad (3.2)$$

where

$$V = \frac{(r + \frac{\Delta r}{4}) \Delta r \Delta \phi \Delta z}{4} \quad (3.3)$$

$$A_o = \frac{(r + \frac{\Delta r}{2}) \Delta \phi \Delta z}{2} \quad (3.4)$$

$$A_\phi = \frac{\Delta r \Delta z}{4} \quad (3.5)$$

$$A_z = \frac{(r + \frac{\Delta r}{4}) \Delta r \Delta \phi}{2} \quad (3.6)$$

We note in Equation 3.2 that the conduction terms are evaluated at time level  $n$  in order to allow explicit solution of the temperatures. Evaluation of  $E_i$  and  $E_j$  reveals that these terms are comprised of the sum of the incident surface heat flux and the terms associated with heat loss (or gain) by convection and radiation. The convection term is linear and may be approximated as

$$q''_c = h_c(T^{n+1} - T_\infty) \quad (3.7)$$

where  $h_c$  is the convection coefficient at the appropriate surface and  $T_\infty$  is the temperature of the surrounding fluid. The radiation term is non-linear and cannot be directly applied as a function of  $T^{n+1}$ . A linearization of the radiation equation may be accomplished by defining the radiation coefficient at the previous ( $n$ ) timestep

$$h_r = \epsilon \sigma (T^n + T_{sw}) [(T^n)^2 + T_{sw}^2] \quad (3.8)$$

where

$$q''_r = h_r (T^{n+1} - T_{sw}) \quad (3.9)$$



which reduces the approximation of the radiation term to the first power of  $T^{n+1}$  and  $T_{sur}$  is the temperature of the surrounding environment to which it radiates. Thus, the boundary conditions may be represented as

$$E_s = A_s \left[ q''_s - h_{c_s} (T^{n+1} - T_\infty) - h_{r_s} (T^{n+1} - T_{sur}) \right] \quad (3.10)$$

and

$$E_l = A_l \left[ q''_l - h_{c_l} (T^{n+1} - T_\infty) - h_{r_l} (T^{n+1} - T_{sur}) \right] \quad (3.11)$$

where

$$A_l = \frac{r \Delta \phi \Delta z}{2} \quad (3.12)$$

Substituting for  $E_s$  and  $E_l$  into Equation 3.2 yields

$$\begin{aligned} \rho C_p V \frac{(T^{n+1} - T^n)}{\Delta t} = & k_z A_z \frac{(T_N^n - T^n)}{\Delta z} + k_\phi A_\phi \frac{(T_E^n + T_W^n - 2T^n)}{r \Delta \phi} \quad (3.13) \\ & + k_r A_o \frac{(T_o^n - T^n)}{\Delta r} + A_s \left[ q''_s + h_{c_s} (T_\infty - T^{n+1}) + h_{r_s} (T_{sur} - T^{n+1}) \right] \\ & + A_l \left[ q''_l + h_{c_l} (T_\infty - T^{n+1}) + h_{r_l} (T_{sur} - T^{n+1}) \right] \end{aligned}$$

Substituting for the volume and area terms yields

$$\begin{aligned}
 \rho C_p \frac{(r + \frac{\Delta r}{2}) \Delta r \Delta \phi \Delta z (T^{n+1} - T^n)}{4 \Delta t} &= \frac{k_z (r + \frac{\Delta r}{4}) \Delta r \Delta \phi (T_N^n - T^n)}{2 \Delta z} \\
 &+ \frac{k_\phi \Delta r \Delta z (T_E^n + T_W^n - 2T^n)}{4 r \Delta \phi} + \frac{k_r (r + \frac{\Delta r}{2}) \Delta \phi \Delta z (T_O^n - T^n)}{2 \Delta r} \\
 &+ \frac{(r + \frac{\Delta r}{4}) \Delta r \Delta \phi}{2} [q''_s + h_{c_s} (T_\infty - T^{n+1}) + h_{r_s} (T_{sur} - T^{n+1})] \\
 &+ \frac{r \Delta \phi \Delta z}{2} [q''_l + h_{c_l} (T_\infty - T^{n+1}) + h_{r_l} (T_{sur} - T^{n+1})]
 \end{aligned} \tag{3.14}$$

By defining the directional thermal diffusivity

$$\alpha_z = \frac{k_z}{\rho C_p} \tag{3.15}$$

and grid Fourier number

$$Fo = \frac{\alpha \Delta t}{\Delta z^2} \tag{3.16}$$

we may simplify by arbitrarily dividing by the lead coefficient of the z-direction difference approximation to obtain

$$\frac{1}{2Fo} (T^{n+1} - T^n) = (T_N^n - T^n) + \frac{k_p \Delta z^2}{2k_s r (r + \frac{\Delta r}{4}) \Delta \phi^2} (T_E^n + T_W^n - 2T^n) \quad (3.17)$$

$$+ \frac{k_r (r + \frac{\Delta r}{2}) \Delta z^2}{k_s (r + \frac{\Delta r}{4}) \Delta r^2} (T_O^n - T^n) \\ + \frac{\Delta z}{k_s} [q''_s + h_{c_s} (T_\infty - T^{n+1}) + h_{r_s} (T_{sur} - T^{n+1})] \\ + \frac{r \Delta z^2}{k_s (r + \frac{\Delta r}{4}) \Delta r} [q''_I + h_{c_I} (T_\infty - T^{n+1}) + h_{r_I} (T_{sur} - T^{n+1})]$$

Further simplification may be realized by defining the constants

$$c_1 = \frac{k_p \Delta z^2}{2k_s r (r + \frac{\Delta r}{4}) \Delta \phi^2} \quad (3.18)$$

$$c_2 = \frac{k_r (r + \frac{\Delta r}{2}) \Delta z^2}{k_s (r + \frac{\Delta r}{4}) \Delta r^2} \quad (3.19)$$

$$c_3 = \frac{\Delta z}{k_s} \quad (3.19)$$

$$c_4 = \frac{r \Delta z^2}{k_s (r + \frac{\Delta r}{4}) \Delta r} \quad (3.20)$$

Substituting into Equation 3.16 and rearranging to yield

$$\begin{aligned} \frac{1}{2Fo}(T^{n+1} - T^n) &= (T_N^n - T^n) + c_1(T_E^n + T_W^n - 2T^n) + c_2(T_O^n - T^n) \quad (3.21) \\ &+ c_3[q''_s - h_{c_s}(T_\infty - T^{n+1}) - h_{r_s}(T_{sur} - T^{n+1})] \\ &+ c_4[q''_I - h_{c_I}(T_\infty - T^{n+1}) - h_{r_I}(T_{sur} - T^{n+1})] \end{aligned}$$

Expansion and regrouping of terms yields the final form of the node equation for the temperature at the (n+1) timestep

$$\begin{aligned} T^{n+1} &= \frac{1}{c_5} \left\{ T^n + 2Fo \left[ (T_N^n - T^n) + c_1(T_E^n + T_W^n - 2T^n) + c_2(T_O^n - T^n) \right. \right. \quad (3.22) \\ &\quad \left. \left. + c_3(q''_s + h_{c_s}T_\infty + h_{r_s}T_{sur}) \right. \right. \\ &\quad \left. \left. + c_4(q''_I + h_{c_I}T_\infty + h_{r_I}T_{sur}) \right] \right\} \end{aligned}$$

where

$$c_5 = 1 + 2Fo \left[ c_3(h_{c_s} + h_{r_s}) + c_4(h_{c_I} + h_{r_I}) \right] \quad (3.23)$$

Thus, the temperature for each node at the successive time step may be solved for explicitly in terms of node temperatures at the previous timestep.

## C. BRIAN'S METHOD

### 1. Step One (z-Direction)

This section addresses procedures governing the development of equations associated with the first intermediate, or *star-level*.

temperature array to which the z-coordinate direction has been arbitrarily assigned. Equation 3.21 may be applied to step one by substituting asterisk-superscripted variables for all n+1 superscripted variables and those n-superscripted variables which lie along the z-coordinate direction. Because the intermediate time step is half the overall value, the effective grid Fourier Number is half that employed in the explicit method. Thus, for step one of Brian's Method, Equation 3.21 becomes

$$\begin{aligned} \frac{1}{Fo} (T^* - T^n) = & (T_N^* - T^n) + c_1 (T_E^n + T_W^n - 2T^n) + c_2 (T_O^n - T^n) \quad (3.24) \\ & + c_3 [q''_s - h_{c_s} (T^* - T_\infty) - h_{r_s} (T^* - T_{sw})] \\ & + c_4 [q''_l - h_{c_l} (T^* - T_\infty) - h_{r_l} (T^* - T_{sw})] \end{aligned}$$

Expanding and grouping like terms of starred quantities on the left-hand side of the equation yields

$$AT_s^* + BT^* + CT_N^* = D \quad (3.25)$$

where

$$A = 0 \quad (3.26)$$

$$B = 1 + Fo [1 + c_3 (h_{c_s} + h_{r_s}) + c_4 (h_{c_l} + h_{r_l})] \quad (3.27)$$

$$C = -Fo \quad (3.28)$$

$$D = T^* + Fo [c_1(T_E^* + T_W^* - 2T^*) + c_2(T_O^* - T^*)] \quad (3.29)$$

$$+ c_3(q''_s + h_{c_s}T_\infty + h_{r_s}T_{sw})$$

$$+ c_4(q''_l + h_{c_l}T_\infty + h_{r_l}T_{sw})]$$

This process is repeated until similar equations are derived for each node type.

## 2. Step Two ( $\phi$ -Direction)

Derivation of step two equations is an extension of the procedures employed in the preceding section with the  $\phi$ -axis now arbitrarily designated the *two-star* coordinate direction. In this step, beginning with Equation 3.24, which already contains the initial step one substitutions, we add further modifications by substituting two-star subscripted variables in the same manner as before. This yields

$$\frac{1}{Fo}(T^{**} - T^*) = (T_N^* - T^*) + c_1(T_E^{**} + T_W^{**} - 2T^{**}) + c_2(T_O^* - T^*) \quad (3.30)$$

$$+ c_3[q''_s - h_{c_s}(T^{**} - T_\infty) - h_{r_s}(T^{**} - T_{sw})]$$

$$+ c_4[q''_l - h_{c_l}(T^{**} - T_\infty) - h_{r_l}(T^{**} - T_{sw})]$$

Again, expanding and grouping like terms of two-starred quantities on the left-hand side of the equation yields

$$AT_E^{**} + BT_W^{**} + CT_W^{**} = D \quad (3.31)$$

where

$$A = - c_1 Fo \quad (3.31)$$

$$B = 1 + Fo [2c_1 + c_2(h_{c_s} + h_{r_s}) + c_3(h_{c_l} + h_{r_l})] \quad (3.32)$$

$$C = - c_1 Fo \quad (3.33)$$

$$D = T'' + Fo [(T_N^* - T^*) + c_2(T_O'' - T'')] \quad (3.34)$$

$$+ c_3 [q''_s + h_{c_s} T_\infty + h_{r_s} T_{sw}]$$

$$+ c_4 (q''_l + h_{c_l} T_\infty + h_{r_l} T_{sw})]$$

This process is again repeated until similar equations are derived for each node type.

### 3. Step Three (r-Direction)

Derivation of step three equations is an identical extension of the procedures employed in the preceding section, with the r-axis now arbitrarily designated the *three-star* coordinate direction. In this step, we begin with Equation 3.30 and further modify by substituting three-star subscripted variables. This yields

$$\frac{1}{Fo}(T^{***} - T^*) = (T_N^* - T^*) + c_1(T_E^{**} + T_W^{**} - 2T^{**}) + c_2(T_O^{***} - T^{***}) \quad (3.35)$$

$$+ c_3[q''_s - h_{c_s}(T^{***} - T_\infty) - h_{r_s}(T^{***} - T_{sur})]$$

$$+ c_4[q''_l - h_{c_l}(T^{***} - T_\infty) - h_{r_l}(T^{***} - T_{sur})]$$

Again, expanding and grouping like terms of two-starred quantities on the left-hand side of the equation yields

$$AT_E^{***} + BT^{***} + CT_W^{***} = D \quad (3.36)$$

where

$$A \approx 0 \quad (3.37)$$

$$B = 1 + Fo[c_2 + c_3(h_{c_s} + h_{r_s}) + c_4(h_{c_l} + h_{r_l})] \quad (3.38)$$

$$C = -c_2 Fo \quad (3.39)$$

$$D = T^* + Fo[(T_N^* - T^*) + c_1(T_E^{**} + T_W^{**} - 2T^{**})] \quad (3.40)$$

$$+ c_3(q''_s + h_{c_s} T_\infty + h_{r_s} T_{sur})$$

$$+ c_4(q''_l + h_{c_l} T_\infty + h_{r_l} T_{sur})]$$

This process is repeated until similar equations are derived for each node type.



#### 4. Step Four (New Temperatures)

Step four is a direct extension of the procedures employed in the explicit method. In this step, the  $n+1$  timestep equations are derived from the explicit equations by substituting starred values for the associated node temperatures for node under consideration. Continuing with our example, we substitute into Equation 3.22 to obtain

$$\begin{aligned}
 T^{n+1} = \frac{1}{c_5} \{ & T^n + 2Fo [(T_N^* - T^*) + c_1(T_E^{**} + T_W^{**} - 2T^{**}) + c_2(T_O^{***} - T^{***}) \quad (3.41) \\
 & + c_3(q''_s + h_{c_s}T_\infty + h_{r_s}T_{sur}) \\
 & + c_4(q''_l - h_{c_l}T_\infty - h_{r_l}T_{sur})] \}
 \end{aligned}$$

where

$$c_5 = 1 + 2Fo[c_3(h_{c_s} + h_{r_s}) + c_4(h_{c_l} + h_{r_l})] \quad (3.42)$$

Step four equations for each of the six types of nodes are derived in an identical fashion.

Complete node equations in the polar-cylindrical form of Brian's method are included in Appendix B for each of the six node types identified in Figure 2.

## **IV. CODE DEVELOPMENT**

### **A ANALYTIC SOLUTION**

Analytic solutions to the model described in Chapter III, paragraph A.2, were obtained for the range of zero to 300 seconds in five-second increments. The calculations of eigenvalues and the resulting solutions were made using MATHCAD V2.0 on an INTEL 80386-based personal computer. The calculated eigenvalues were verified with tabular results [Ref. 5] and those obtained by an appropriate IMSL subroutine [Refs. 6-10]. Equations describing the complete solution are contained in Appendix C.

### **B EXPLICIT METHOD**

#### **1. Requirements**

The coding of the explicit method is relatively straightforward and subject to only two notable requirements:

- That the selected timestep must not exceed the theoretical maximum, and
- That each node be considered for temperature calculation once per timestep.

#### **2. General Procedure**

##### **a Timestep Restriction**

The maximum allowable timestep varies with control volume dimensions. Exceeding the allowable timestep for *any one node* will result in the eventual loss of stability of the entire system. The rapidity with which this occurs is a function of the affected node's

significance in the path of the dominating heat flux. Consequently, a maximum timestep must be determined for each node type (Figure 2) and the most restrictive taken as the limiting value. This was effected through use of the subroutine *STABIL*, which calculated maximum allowable timestep for each node-type and returned the most restrictive as the argument *DTMAX*. A listing of equations defining stability requirements for each node type is contained as Appendix D.

#### ***b. Indexing***

The method by which incrementation through the grid system is implemented is entirely arbitrary as long as the temperature is calculated once per timestep for each node on the grid. In systems with a dominant heat flux, convergence will occur more quickly if the dominant direction is incremented first. Appendix E contains the explicit method code as developed for this study. A polar-cylindrical coordinate system was arbitrarily applied with the z-axis aligned with the central axis of the cylinder. Indexing commenced at the inner edge of the bottom surface and proceeded up the z-coordinate to the upper surface. The grid was then indexed in  $\phi$ , followed by r, until the entire grid had been covered, at which time the old temperature matrix was replaced by the new temperature matrix, the time counter indexed by  $\Delta t$ , and the process repeated until the end of the desired period of evaluation was reached.

## **C. BRIAN'S METHOD**

### **1. Requirements**

#### **a Timestep**

The timestep may be selected as befits the duration of the analysis and the expected temperature gradients. However, the discretization error is a function of the square of the timestep. On the other hand, the property of unconditional stability allows the flexibility to initially choose a large timestep for macro-analysis followed by application of a smaller timestep for a detailed look at regions of interest (e.g., an area where a predefined threshold is exceeded).

#### **b. Indexing**

A separate temperature matrix must be developed for each coordinate direction at the level of the intermediate timestep. Due to this directional nature, the indexing requirements are more complex and restrictive. Once assignment of a coordinate axis to an intermediate matrix (e.g., star-level) has been made, the convention must be adhered to for the duration of the timestep. One acceptable method for indexing is outlined below and the companion code is included as the *STEP(#)* subroutines to the Brian's Method code contained in Appendix F.

**(1) Star Level (z-Direction).** The z-coordinate direction was arbitrarily assigned to the star-level intermediate matrix. Consequently, equations were developed for nodes lying along the z-axis in the fashion described in Section C of Chapter III. Appropriate indexing was accomplished by designating an arbitrary node on the

inner ring of the bottom surface of the cylinder (Figure 7) as a node of reference (node 1). An equation similar to Equation 3.25 was developed for this first node and the values of the A, B, C, and D coefficients calculated and stored in four associated vectors. The pointer was then indexed in the z-direction to select the node immediately above node 1 (i.e., the northern neighbor when developing the equation for node 1). The process was repeated for the second node and all other nodes lying on the selected z-axis

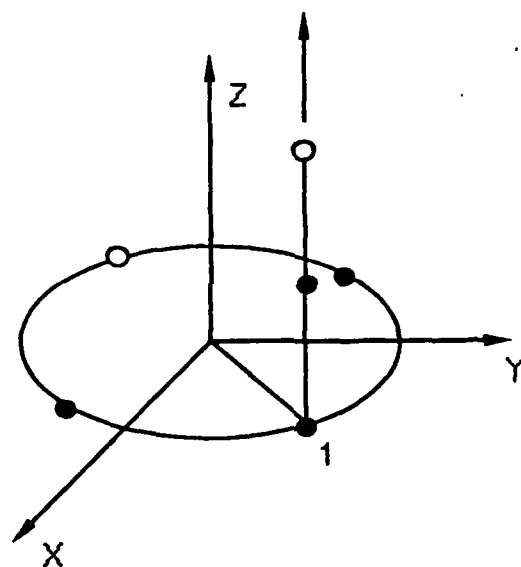


Figure 7. **Star-Level Indexing Direction**

When the values of the coefficients for the last node had been calculated and stored in the appropriate vectors, a tridiagonal matrix solver was called to simultaneously solve the system of equations. The resultant node temperatures were then saved in the star-level intermediate matrix and the pointer indexed to the next adjacent node in the

counter-clockwise  $\phi$ -direction. Note that at this point, indexing in either the radial or the  $\phi$ -direction would have been equally appropriate. However, once chosen, the specified direction must be adhered to throughout the immediate timestep.

**(2) Two-Star Level ( $\phi$ -Direction).** Because the rings associated with the  $\phi$ -direction have neither a beginning nor an end, one node must be arbitrarily defined as a reference node. Using node 1 from the one-star level and working in a counter-clockwise direction (Figure 8), the values of the coefficients for each node along the chosen ring were calculated and temporarily stored in suitable vectors. The resultant system of equations for nodes in the  $\phi$ -direction was of the form

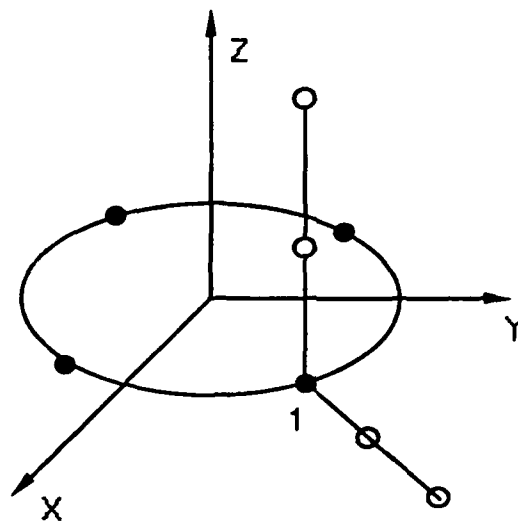


Figure 8. **Two-Star Level Indexing**

$$\begin{bmatrix} b_1 & c_1 & 0 & a_1 \\ a_2 & b_2 & c_2 & 0 \\ 0 & a_3 & b_3 & c_3 \\ c_3 & 0 & a_3 & b_4 \end{bmatrix} \cdot \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} \quad (4.1)$$

which is not tridiagonal. To achieve tridiagonality, the ring was cut at the reference node. The resulting coefficient matrix was then of the form

$$\begin{bmatrix} (b_1 - a_1) & c_1 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 \\ 0 & a_3 & b_3 & c_3 \\ 0 & 0 & a_4 & (b_4 - c_4) \end{bmatrix} \quad (4.2)$$

Node temperatures were then obtained by applying a suitable tridiagonal matrix solver [Ref. 4:p. 441] and stored in the two-star intermediate level matrix. The pointer was then incremented to the node adjacent to node 1 in the r-direction and the process repeated. Upon completion of the first layer calculations, the pointer was indexed in the positive z-direction and the process repeated until all two-star level temperatures have been calculated.

**(3) Three-Star Level (r-Direction).** Construction of the three-star temperature matrix was accomplished in a fashion similar to the one- and two-star level procedures (Figure 9). Beginning at the reference node, the values of the node coefficients were calculated and temporarily stored in suitable vectors. When the coefficients for the last node on the selected ray had been calculated and stored, the tridiagonal matrix solver was applied to the system of equations, the

resultant node temperatures written to the three-star temperature array, the pointer indexed to the next adjacent ray in the counter-clockwise direction, and the process repeated until all node temperatures were calculated for the layer. The pointer was then indexed in the positive  $z$ -direction until all three-star level temperatures had been calculated.

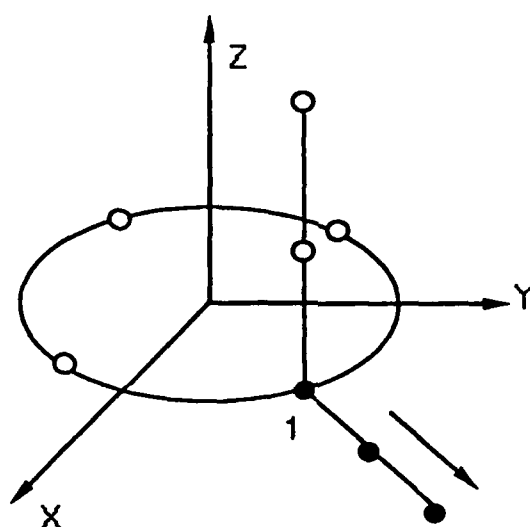


Figure 9. **Three-Star Indexing**

**(4) New Temperature Array ( $n+1$  Level).** When all intermediate matrices had been developed, new node temperatures were calculated in terms of the three intermediate temperature arrays and the array of temperatures at the previous (old) timestep in accordance with the discussion in paragraph 4 of Chapter III. The choice of indexing direction at this point was arbitrary, so the method applied to the star level was repeated. It must be noted that *the new node*



*temperatures must be stored in a separate (new-temperature) array until all node temperatures have been calculated. Upon completion of the calculations, the new temperature array was printed or stored (when appropriate), copied onto the old temperature array, the time counter indexed, and the entire process repeated until the limit of the period of evaluation was reached.*

#### **D. EXECUTION TIMEKEEPING**

The codes for both the explicit method and Brian's Method were of common origin, had nearly identical main codes, and shared common input/output and utility subroutines. The codes were organized such that the calculations unique to the specific method employed were segregated into a single, cohesive block in the form of one or more subroutines. A timing utility which allowed placement within the code of pointers to initiate and stop CPU timekeeping was employed to measure the execution times of each method for each example or application tested. Among other places, pointers were placed at the entry to and the exit from the respective method-specific subroutine blocks. Benchmark runs exclusive of I/O calls were conducted for each method during the validation process. Subsequent runs involving printed or stored output were structured such that required I/O handling instructions were common to each code to allow accurate comparison of relative speeds.

## **V. VALIDATION**

### **A INITIAL CODE**

#### **1. General Procedure**

To simplify code development, initial forms of the codes for both the explicit method and Brian's Method were developed without including convection or radiation. Validation for both methods was accomplished by solving for the temperature distribution within the cylinder described in paragraph A.2 of Chapter III, comparing results with the analytic solution and calculating the percentage error. Upon achievement of satisfactory results, the initial codes were expanded to encompass the effects of both convection and radiation. Validation of the core of this revised code was accomplished by setting appropriate convective and radiative coefficients to zero and repeating the initial validation procedure with the revised code.

#### **2. Time Incrementation**

Initial runs were conducted using a fixed time step of 0.1 second. Following initial data collection, additional runs were conducted to determine operating limits for both methods. During this phase, Brian's method was tested and validated using constant time increments in the range from 0.01 to 100 seconds and included a logarithmically increasing time step arrangement summarized in Table 3.

TABLE 3  
VARIABLE TIME STEP EMPLOYMENT

RANGE (SEC)	TIME STEP (SEC)
0.1-1.0	0.1
1.0-10.0	1.0
10.0-100	10.0
100-300	100.0

Additional runs were conducted using a varying time step in which the final step size was limited to 20 seconds.

#### **B. APPLICATION CODE**

No analytic solution was available for a cylinder with convective and radiative boundary conditions. Consequently, the application codes may not be considered fully validated. A comparison of results was obtained by solving the application example with both methods and comparing the results point-for-point against each other in the form of a percentage disagreement. All combinations of heat flux and boundary conditions considered are summarized in Table 4.

TABLE 4  
SUMMARY OF APPLIED BOUNDARY CONDITIONS

Boundary Class	Inner	Outer	Upper	Lower
I	Q	R	R	R
II	Q	R,Q	R	R
III	Q	R,Q	R,Q	R
IV	Q	R,Q	R	R,Q
V	Q	R,Q	R,Q	R,Q
VI	Q	R,Q,H	R	R
VII	Q	R,Q	R,H	R
VIII	Q	R,Q	R	R,H
IX	Q	R,Q	R,H	R,H
X	Q	R,Q,V1	R	R
XI	Q	R,Q	R,V2	R
XII	Q	R,Q	R	R,V3
XIII	Q	R,Q	R,V2	R,V3
XIV	Q	R,Q,V1	R,V2	R,V3
XV	Q	R,Q	R,V2	R,V2
XVI	Q	R,Q	R,V3	R,V3

Where:

Q = Constant, uniform surface heat flux

R = Radiation to extensive, black surroundings

H = Convection from surface

V1\* = Periodic, uniform surface heat flux 1

V2\* = Periodic, uniform surface heat flux 2

V3\* = Periodic, uniform surface heat flux 3

\* As described in Appendix G.

## **VI. RESULTS**

### **A. DEFINITIONS**

#### **1. Convergence**

On validation runs, calculated values were considered to have converged when the net remaining change in the value comprized less than one percent of the analytic solution for the given time step. For comparison runs, values were considered to have converged when their difference comprized less than one percent of the value of the smaller of the two numbers.

#### **2. Steady-State**

A condition of steady-state was considered to have been reached when the net remaining change in a temperature comprized less than one percent of the final steady-state result for single transient evaluation.

### **B. VALIDATION**

#### **1. Analytic Solution**

The analytic solution converged to steady-state at 300 seconds. Consequently, all validation runs were limited to 300 seconds duration to economize on analysis time.

#### **2. Explicit Method**

The theoretical limit on the maximum time step allowed for the geometry selected was approximately 0.395 seconds, which did not lend itself to convenient comparisons with the analytic results.

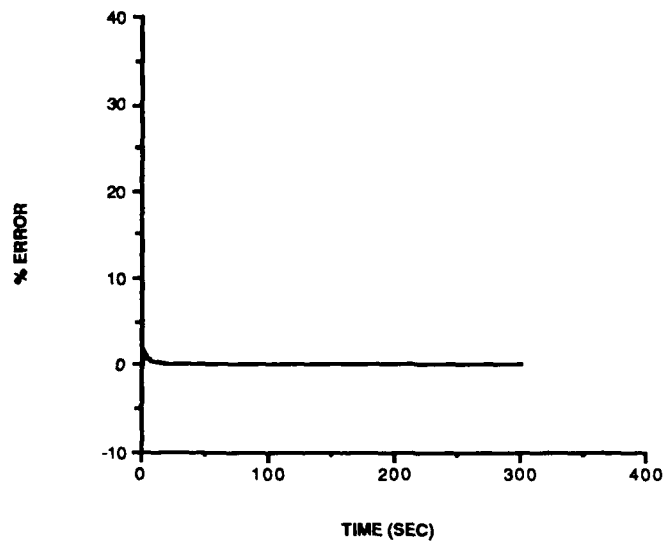
The explicit method was validated using a numerically convenient time step of 0.1000 second and converged with the analytic result within four seconds of the start of the transient cycle (Figure 10). Total execution time was 40.37 seconds for a ten (radial) by four ( $\phi$ ) by five (z) grid. The observed error at the end of the 300-second evaluation period was on the order of 0.009 percent. For processing speed comparison runs, the maximum theoretical timestep was employed and demonstrated a minimum execution time of 10.53 CPU seconds. Due to the odd step size, no comparison of relative accuracy was made.

### **3. Brian's Method**

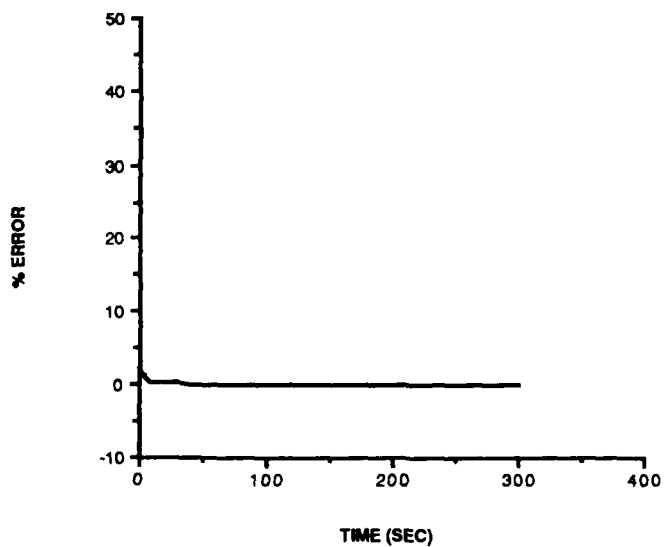
Brian's method was initially limited to a fixed timestep of 0.1 second for the validation process to allow a relative speed comparison with the explicit method. While convergence times and observed errors were comparable (Figures 10, 11, and 12), the explicit method demonstrated a significant speed advantage over Brian's Method for equivalent timesteps (Table 5). This situation was dramatically reversed when the logarithmically increasing timestep arrangement (Table 3) was employed and resulted in an execution speed of 2.53 CPU seconds, which comprised a 410 percent advantage over the optimum result of the explicit method (Table 6).

### **C. FOLLOW-ON TESTING**

Upon conclusion of the validation runs and speed comparisons, a variety of boundary conditions with easily predictable effects were applied to allow observation of system response (Table 4). The following general observations were made.



**Figure 10. Percentage Error of Explicit Code as Compared to Analytic Results and Plotted Versus Time**



**Figure 11. Percentage Error of Brian's Code as Compared to Analytic Results and Plotted Versus Time**

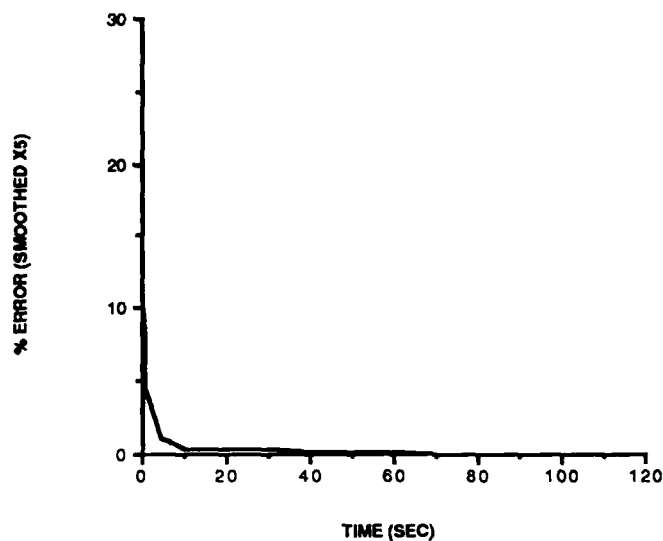


Figure 12. **Comparison of Brian vs. Explicit Validation Results  
Expressed as a Percentage Difference**

TABLE 5

**VALIDATION RESULTS—FIXED 0.1 SEC. TIMESTEP**

10 x 4 x 5 GRID  
300 SEC TRANSIENT

RESULT	EXPLICIT	BRIAN'S
STEP SIZE	0.1 SEC	0.1 SEC
CPU TIME	40.37 SEC	223.24 SEC
ERROR AFTER		
0.1%	15.188%	20.308%
1%	1.247%	1.668%
2%	0.612%	0.814%
10%	0.110%	0.183%
100%	0.009%	0.009%
CONVERGENCE (<1 % ERROR)	3-4 SEC	4-5 SEC

SPEED ADVANTAGE: EXPLICIT (553 %)



TABLE 6  
**VALIDATION RESULTS—LOG-VARIABLE TIMESTEP**

BRIAN'S METHOD  
 VARIABLE STEP SIZE

TIME (SEC)		STEP SIZE (SEC)	
0.1-1.0		0.1	
1.0-10		1.0	
10-100		10	
100-300		20	
% TRANSIENT		% ERROR	% IMPROVEMENT*
0.1		20.308	0.00
1.0		1.446	13.3
10.		-0.057	68.8
100.		0.01	-22.2

CPU TIME: 2.53 SEC

SPEED ADVANTAGE : BRIAN'S (416 %)
-----------------------------------

\* Explicit method with optimum time step.

**1. Symmetric Boundary Conditions**

For both radially and axi-symmetric boundary conditions, the results of both methods rapidly converged with each other within the first three percent of the transient cycle.

**2. Asymmetric Boundary Conditions**

When asymmetric boundary conditions were applied, large differences were observed at surfaces nearest the surface of application. These differences were eliminated by application of a finer grid

size in the direction of principal flux and are thus attributed to discretization errors due to the generally coarse nature of the applied grid.

#### **D. APPLICATION EXAMPLE**

A satellite thermal analysis problem (Appendix G) was selected as offering a practical example of a system subjected to long-period steady-state transient behavior. Figure 13 presents a graphical comparison of the results of both methods for the initial 3000 seconds of the first transient cycle. Figure 14 exhibits the thermal response of selected nodes as calculated by Brian's Method for the initial 10 transient cycles of the application example.

# SATELLITE TEMPERATURE PROFILE (BRIAN VS EXPLICIT)

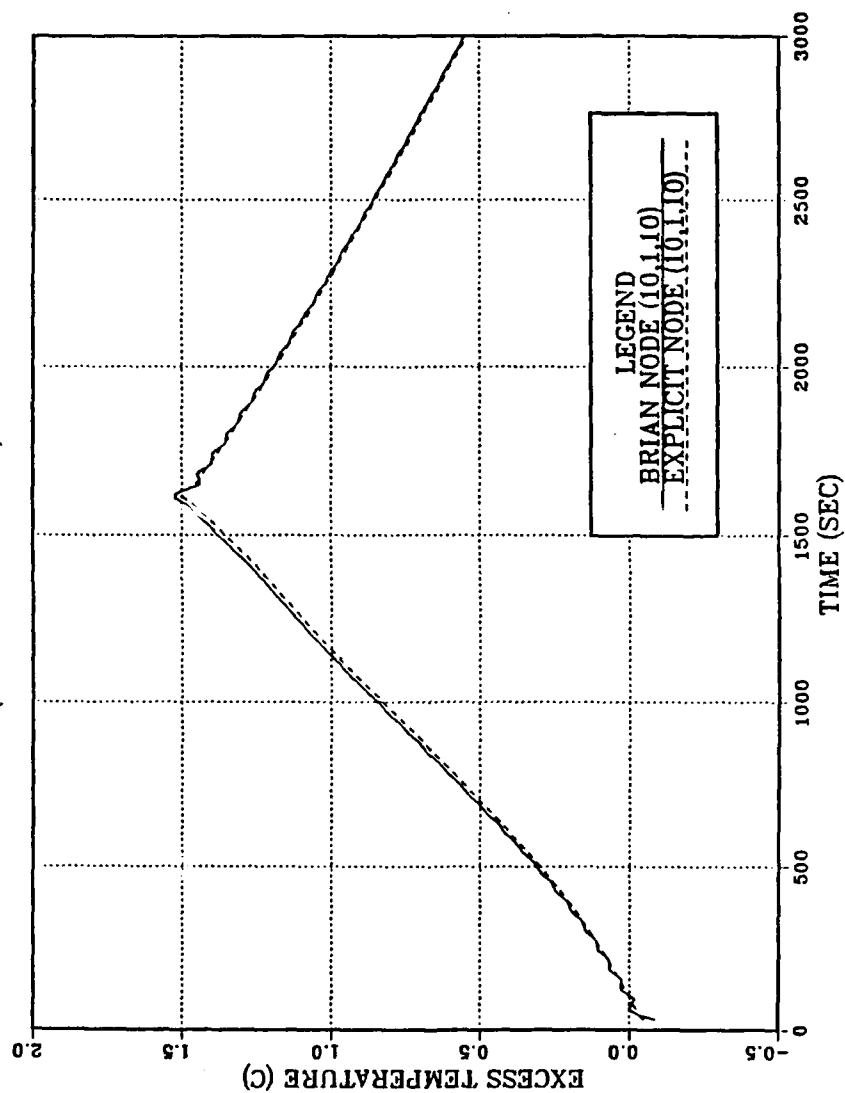


Figure 13. Graphical Comparison of Brian's Method and the Explicit Method for the Initial 3,000 Seconds of the First Transient Cycle of the Application Problem

# SATELLITE TEMPERATURE PROFILE (BRIAN - 10 ORBIT)

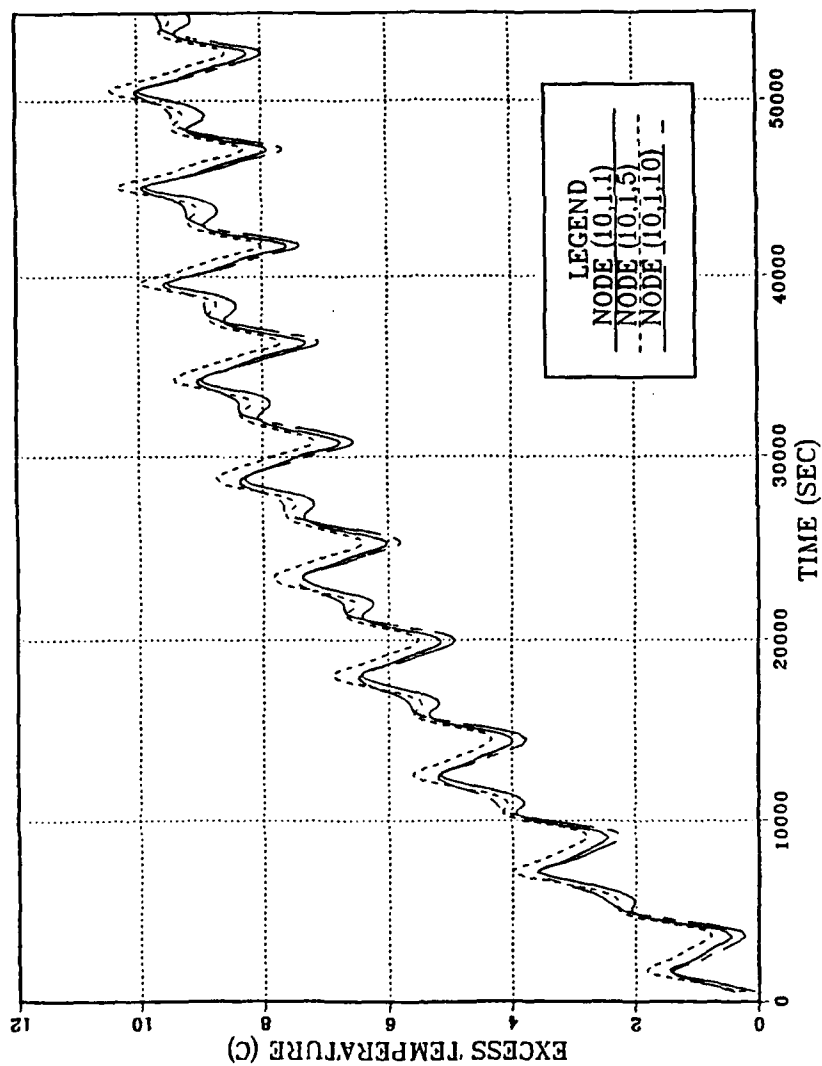


Figure 14. Thermal Response of Selected Nodes as Calculated by Brian's Method for the Initial 10 Transient Cycles of the Application Problem

## **VII. CONCLUSIONS AND RECOMMENDATIONS**

### **A. CONCLUSIONS**

#### **1. Adaptability**

Brian's Method is readily adaptable to cylindrical geometries. However, compensation must be made for the lack of tridiagonality of the coefficient matrix when calculating intermediate temperatures in the  $\phi$ -direction.

#### **2. Relative Accuracy**

For symmetric boundary conditions, Brian's Method delivers accuracy comparable to the explicit method. Convergence is negligibly slower than that observed for the explicit method.

#### **3. Relative Execution Speed**

Brian's Method offers distinct advantages over the explicit method for systems demonstrating periodic steady-state temperature response, and for those systems with extended single transient cycles whose geometry makes the explicit timestep limitation overly restrictive. For some geometries, Brian's method is slower than the explicit method and is not recommended.

#### **4. Relative Benefits**

The principal benefit of Brian's Method is its unconditional stability and associated freedom from timestep restrictions. This allows the user to apply a coarse grid and timestep to allow rapid coarse analysis with the ability to adjust the step size downward as the

situation warrants. However, the relative advantages of the unrestricted timestep must be carefully considered against the substantial increase in algebra and code requirements.

## **B. RECOMMENDATIONS**

### **1. Adaptation of Brian's Method FORTRAN Code to C Programming Language**

#### ***a. Background***

The large storage and calculation requirements associated with finite difference approximations of transient heat conduction problems have historically relegated their analysis to the realm of the mainframe computer. However, recent developments in microprocessor technology have dramatically improved the storage capability and execution speeds of personal computers (PCs). Consequently, increasing amounts of engineering analysis and design are being conducted on these devices at significant cost reductions. In research being conducted under the auspices of the Electrical and Computer Science Department at the Naval Postgraduate School, FORTRAN codes involving large numbers of repetitive calculations or iterations have been demonstrated to experience significant improvements in execution speed when converted to the C programming language prior to execution on a personal computer [Ref. 11].

#### ***b. Recommendation***

It is strongly recommended that research be conducted to investigate the feasibility of and limitations on the conduct of transient heat conduction analysis on personal computers employing a C

programming language adaptation of the Brian's Method FORTRAN code.

**2. Incorporation of the Analysis of Directional and Time-Dependent Properties in Brian's Method**

***a. Background***

Composites demonstrate strong directional and temperature dependent thermal properties and are being rapidly assimilated into many engineering applications.

***b. Recommendation***

It is recommended that additional research be conducted to develop procedures for the incorporation of directional and time-dependent thermal properties into Brian's Method.

## APPENDIX A

### NODE EQUATIONS FOR EXPLICIT METHOD

#### A TYPE 1 NODES—END LAYERS, INNER SURFACE

$$\begin{aligned}
 T^{n+1} = \frac{1}{c_5} \{ & T^n + 2Fo \left[ (T_s^n + T^n) + c_1 (T_E^n + T_w^n - 2T^n) \right. \\
 & + c_2 (T_o^n - T^n) \\
 & + c_3 (q''_e + h_c T_\infty + h_r T_{sur}) \\
 & \left. + c_4 (q''_i + h_c T_\infty + h_r T_{sur}) \right] \}
 \end{aligned}$$

where

$$c_1 = \frac{k_s \Delta z^2}{2 k_i r (r + \frac{\Delta r}{4}) \Delta \phi^2}$$

$$c_2 = \frac{k_r (r + \frac{\Delta r}{2}) \Delta z^2}{k_i (r + \frac{\Delta r}{4}) \Delta r^2}$$

$$c_3 = \frac{\Delta z}{k_i}$$

$$c_4 = \frac{r \Delta z^2}{k_i (r + \frac{\Delta r}{4}) \Delta r}$$



$$c_5 = 1 + 2Fo \left[ c_3(h_{c_e} + h_{r_e}) + c_4(h_{c_i} + h_{r_i}) \right]$$

$$h_{r_e} = \epsilon_e \sigma (T^n + T_{sw}) \left[ (T^n)^2 + T_{sw}^2 \right]$$

$$h_{r_i} = \epsilon_i \sigma (T^n + T_{sw}) \left[ (T^n)^2 + T_{sw}^2 \right]$$

## B TYPE 2 NODES—END LAYERS, MID-RADI

$$\begin{aligned} T^{n+1} = \frac{1}{c_5} \{ & T^n + 2Fo \left[ (T_e^n - T^n) + c_1(T_E^n + T_W^n - 2T^n) \right. \\ & + c_2(T_I^n - T^n) \\ & + c_3(T_O^n - T^n) \\ & \left. + c_4(q''_e + h_{c_e}T_\infty + h_{r_e}T_{sw}) \right] \} \end{aligned}$$

where

$$c_1 = \frac{k_\phi \Delta z^2}{2k_z r^2 \phi^2}$$

$$c_2 = \frac{k_r(r - \frac{\Delta r}{2})\Delta z^2}{2k_z r \Delta r^2}$$

$$c_3 = \frac{k_r(r + \frac{\Delta r}{2})\Delta z^2}{2k_z r \Delta r^2}$$

$$c_4 = \frac{\Delta z}{k_z}$$

$$c_5 = 1 + 2Fo \left[ c_4(h_{c_e} + h_{r_e}) \right]$$

$$h_{r_e} = \epsilon_e \sigma (T^n + T_{sur}) [(T^n)^2 + T_{sur}^2]$$

### C. TYPE 3 NODES—END LAYERS, OUTER SURFACE

$$\begin{aligned} T^{n+1} = \frac{1}{c_5} \{ & T^n + 2Fo \left[ (T_e^n - T^n) + c_1 (T_E^n + T_W^n - 2T^n) \right. \\ & + c_2 (T_I^n - T^n) \\ & + c_3 (q''_e + h_{c_e} T_\infty + h_{r_e} T_{sur}) \\ & \left. + c_4 (q''_o + h_{c_o} T_\infty + h_{r_o} T_{sur}) \right] \} \end{aligned}$$

where

$$c_1 = \frac{k_\phi \Delta z^2}{2k_z \left(r - \frac{\Delta r}{4}\right) r \Delta \phi^2}$$

$$c_2 = \frac{k_r \left(r - \frac{\Delta r}{2}\right) \Delta z^2}{k_z \left(r - \frac{\Delta r}{4}\right) \Delta r^2}$$

$$c_3 = \frac{\Delta z}{k_z}$$

$$c_4 = \frac{r \Delta z^2}{k_z (r - \frac{\Delta r}{4}) \Delta r}$$

$$c_5 = 1 + 2Fo \left[ c_3(h_{c_e} + h_{r_e}) + c_4(h_{c_o} + h_{r_o}) \right]$$

$$h_{r_e} = \epsilon_e \sigma (T^n + T_{sw}) \left[ (T^n)^2 + T_{sw}^2 \right]$$

$$h_{r_o} = \epsilon_o \sigma (T^n + T_{sw}) \left[ (T^n)^2 + T_{sw}^2 \right]$$

#### D. TYPE 4 NODES—MID-LAYERS, INNER SURFACE

$$\begin{aligned} T^{n+1} = & \frac{1}{c_4} \left\{ T^n + Fo \left[ (T_N^n + T_S^n - 2T^n) + c_1(T_E^n + T_W^n - 2T^n) \right. \right. \\ & + c_2(T_o^n - T^n) \\ & \left. \left. + c_3(q''_I + h_{c_I} T_\infty + h_{r_I} T_{sw}) \right] \right\} \end{aligned}$$

where

$$c_1 = \frac{k_\phi \Delta z^2}{k_z r (r + \frac{\Delta r}{4}) \Delta \phi^2}$$

$$c_2 = \frac{2k_r (r + \frac{\Delta r}{2}) \Delta z^2}{k_z (r + \frac{\Delta r}{4}) \Delta r^2}$$

$$c_3 = \frac{2r \Delta z^2}{k_z(r + \frac{\Delta r}{4}) \Delta r}$$

$$c_4 = 1 + Fo \left[ c_3(h_{c_I} + h_{r_I}) \right]$$

$$h_{r_I} = \epsilon_I \sigma (T^n + T_{sur}) \left[ (T^n)^2 + T_{sur}^2 \right]$$

#### E. TYPE 5 NODES—MID-LAYERS, MID-RADII

$$\begin{aligned} T^{n+1} = & T^n + Fo \left[ (T_N^n + T_S^n - 2T^n) + c_1(T_E^n + T_W^n - 2T^n) \right. \\ & + c_2(T_I^n - T^n) \\ & \left. + c_3(T_O^n - T^n) \right] \end{aligned}$$

where

$$c_1 = \frac{k_\phi \Delta z^2}{k_z r^2 \Delta \phi^2}$$

$$c_2 = \frac{k_r(r + \frac{\Delta r}{2}) \Delta z^2}{k_z r \Delta r^2}$$

$$c_3 = \frac{(r + \frac{\Delta r}{2}) \Delta z^2}{k_z \Delta r^2}$$

# **F. TYPE 6 NODES—MID-LAYERS, OUTER SURFACE**

$$T^{n+1} = \frac{1}{c_4} \left\{ T^n + Fo \left[ (T_N^n + T_S^n - 2T^n) + c_1 (T_E^n + T_W^n - 2T^n) \right. \right. \\ \left. \left. + c_2 (T_I^n - T^n) \right. \right. \\ \left. \left. + c_3 (q''_o + h_{c_o} T_\infty + h_{r_o} T_{sur}) \right] \right\}$$

where

$$c_1 = \frac{k_\phi \Delta z^2}{k_z r (r - \frac{\Delta r}{4}) \Delta \phi^2}$$

$$c_2 = \frac{2k_r (r - \frac{\Delta r}{4}) \Delta z^2}{k_z (r - \frac{\Delta r}{4}) \Delta r^2}$$

$$c_3 = \frac{2r \Delta z^2}{k_z (r - \frac{\Delta r}{4}) \Delta r}$$

$$c_4 = 1 + Fo \left[ c_3 (h_{c_o} + h_{r_o}) \right]$$

$$h_{r_o} = \epsilon_o \sigma (T^n + T_{sur}) \left[ (T^n)^2 + T_{sur}^2 \right]$$

## APPENDIX B

### NODE EQUATIONS FOR BRIAN'S METHOD

#### A TYPE 1 NODES—END LAYERS, INNER SURFACE

$$c_1 = \frac{k_z \Delta z^2}{2k_z r (r + \frac{\Delta r}{4}) \Delta \phi^2}$$

$$c_2 = \frac{k_r (r + \frac{\Delta r}{2}) \Delta z^2}{k_z (r + \frac{\Delta r}{4}) \Delta r^2}$$

$$c_3 = \frac{\Delta z}{k_z}$$

$$c_4 = \frac{r \Delta z^2}{k_z (r + \frac{\Delta r}{4}) \Delta r}$$

$$c_5 = 1 + 2Fo \left[ c_3 (h_{c_e} + h_{r_e}) + c_4 (h_{c_i} + h_{r_i}) \right]$$

$$h_{r_e} = \epsilon_e \sigma (T^n + T_{sw}) [(T^n)^2 + T_{sw}^2]$$

$$h_{r_i} = \epsilon_i \sigma (T^n + T_{sw}) [(T^n)^2 + T_{sw}^2]$$

### 1. Step 1 (z-Direction)

$$AT_s^* + BT^* + CT_N^* = D$$

where

$$A = \begin{cases} 0, & \text{Bottom layer nodes} \\ -Fo, & \text{Top layer nodes} \end{cases}$$

$$B = 1 + Fo \left[ 1 + c_2(h_{c_e} + h_{r_e}) + c_3(h_{c_i} + h_{r_i}) \right]$$

$$C = \begin{cases} -Fo, & \text{Top layer nodes} \\ 0, & \text{Bottom layer nodes} \end{cases}$$

$$D = T^* + Fo \left[ c_1(T_E^* + T_w^* - 2T^*) + c_2(T_o^* - T^*) \right.$$

$$+ c_3(q''_e + h_{c_e}T_\infty + h_{r_e}T_{sur})$$

$$\left. + c_4(q''_i + h_{c_i}T_\infty + h_{r_i}T_{sur}) \right]$$

### 2. Step 2 ( $\phi$ -Direction)

$$AT_E^{**} + BT^{**} + CT_w^{**} = D$$

where

$$A = -c_1 Fo$$

$$B = 1 + Fo \left[ 2c_1 + c_3(h_{c_e} + h_{r_e}) + c_4(h_{c_l} + h_{r_l}) \right]$$

$$C = -c_1 Fo$$

$$D = T + Fo \left[ (T_e^* - T^*) + c_4(T_o^* - T^*) \right.$$

$$+ c_3(q''_e + h_{c_e} T_\infty + h_{r_e} T_{sur})$$

$$\left. + c_4(q''_l + h_{c_l} T_\infty + h_{r_l} T_{sur}) \right]$$

### 3. Step 3 (r-Direction)

$$AT_l^{***} + BT^{***} + CT_o^{***} = D$$

where

$$A = 0$$

$$B = 1 + Fo \left[ c_2 + c_3(h_{c_e} + h_{r_e}) + c_4(h_{c_l} + h_{r_l}) \right]$$

$$C = -c_2 Fo$$

$$D = T^* + Fo \left[ (T_e^* - T^*) + c_1(T_E^{**} + T_W^{**} - 2T^{**}) \right.$$

$$+ c_3(q''_e + h_{c_e} T_\infty + h_{r_e} T_{sur})$$



$$+ c_4(q''_1 + h_{c_1} T_\infty + h_{r_1} T_{sur})]$$

#### 4. Step 4 ( $\phi$ -Direction)

$$T^{n+1} = \frac{1}{c_5} \left\{ T^n + 2Fo \left[ (T_s^* + T^*) + c_1(T_E^{**} + T_W^{**} - 2T^{**}) \right. \right.$$

$$\left. + c_2(T_O^{***} - T^{***}) \right.$$

$$\left. + c_3(q''_s + h_{c_s} T_\infty + h_{r_s} T_{sur}) \right.$$

$$\left. + c_4(q''_1 - h_{c_1} T_\infty - h_{r_1} T_{sur}) \right\}$$

#### B. TYPE 2 NODES—END LAYERS, MID-RADII

$$c_1 = \frac{k_s \Delta z^2}{2k_r r^2 \phi^2}$$

$$c_2 = \frac{k_r(r - \frac{\Delta r}{2}) \Delta z^2}{2k_s r \Delta r^2}$$

$$c_3 = \frac{k_r(r + \frac{\Delta r}{2}) \Delta z^2}{2k_s r \Delta r^2}$$

$$c_4 = \frac{\Delta z}{k_s}$$

$$c_5 = 1 + 2Fo [c_4(h_{c_s} + h_{r_s})]$$

$$h_{r,e} = \varepsilon_e \sigma (T^n + T_{sw}) \left[ (T^n)^2 + T_{sw}^2 \right]$$

### 1. Step 1 (z-Direction)

$$AT_s^* + BT^* + CT_N^*$$

where

$$A = \begin{cases} 0, & \text{Bottom layer nodes} \\ -Fo, & \text{Top layer nodes} \end{cases}$$

$$B = 1 + Fo \left[ 1 + c_4 (h_{c,e} + h_{r,e}) \right]$$

$$C = \begin{cases} -Fo, & \text{Top layer nodes} \\ 0, & \text{Bottom layer nodes} \end{cases}$$

$$\begin{aligned} D = T^n + Fo & \left[ c_1 (T_E^n + T_W^n - 2T^n) + c_2 (T_I^n - T^n) \right. \\ & \left. + c_3 (T_O^n - T^n) \right. \\ & \left. + c_4 (q'_{e,e} + h_{c,e} T_{\infty} + h_{r,e} T_{sw}) \right] \end{aligned}$$

### 2. Step 2 ( $\phi$ -Direction)

$$AT_E^{**} + BT^{**} + CT_W^{**} = D$$

where

$$A = -c_1 Fo$$

$$B = 1 + Fo \left[ 2c_1 + c_4(h_{c_e} + h_{r_e}) \right]$$

$$C = - c_1 Fo$$

$$D = T^{\circ} + Fo \left[ (T_e^{\circ} - T^{\circ}) + c_2(T_f^{\circ} - T^{\circ}) + c_3(T_o^{\circ} - T^{\circ}) \right. \\ \left. + c_4(q''_e + h_{c_e} T_{\infty} + h_{r_e} T_{sur}) \right]$$

### 3. Step 3 (r-Direction)

$$AT_f^{***} + BT^{***} + CT_o^{***} = D$$

where

$$A = - c_2 Fo$$

$$B = 1 + Fo \left[ c_2 + c_3 + c_4(h_{c_e} + h_{r_e}) \right]$$

$$C = - c_3 Fo$$

$$D = T^{\circ} + Fo \left[ (T_e^{\circ} - T^{\circ}) + c_1(T_E^{\circ\circ} + T_W^{\circ\circ} - 2T^{\circ\circ}) \right. \\ \left. + c_4(q''_e + h_{c_e} T_{\infty} + h_{r_e} T_{sur}) \right]$$

#### 4. Step 4 (New Temperatures)

$$\begin{aligned}
 T^{n+1} = \frac{1}{c_5} \{ & T^n + 2Fo [(T_s^* - T^*) + c_1(T_E^{**} + T_W^{**} - 2T^{**}) \\
 & + c_2(T_I^{**} - T^{***}) + c_3(T_O^{***} - T^{***}) \\
 & + c_4(q''_s + h_{c,s}T_s + h_{r,s}T_{sw})] \}
 \end{aligned}$$

#### C. TYPE 3 NODES—END LAYERS, OUTER SURFACE

$$c_1 = \frac{k_s \Delta z^2}{2k_z(r - \frac{\Delta r}{4})r\Delta\phi^2}$$

$$c_2 = \frac{k_r(r - \frac{\Delta r}{2})\Delta z^2}{k_z(r - \frac{\Delta r}{4})\Delta r^2}$$

$$c_3 = \frac{\Delta z}{k_z}$$

$$c_4 = \frac{r\Delta z^2}{k_z(r - \frac{\Delta r}{4})\Delta r}$$

$$c_5 = 1 + 2Fo [c_3(h_{c,s} + h_{r,s}) + c_4(h_{c,o} + h_{r,o})]$$

$$h_{r,s} = \epsilon_s \sigma (T^n + T_{sw}) [(T^n)^2 + T_{sw}^2]$$

$$h_{r_o} = \varepsilon_o \sigma (T^{\wedge} + T_{sur}) \left[ (T^{\wedge})^2 + T_{sur}^2 \right]$$

### 1. Step 1 (z-Direction)

$$AT_s^* + BT^* + CT_N^* = D$$

where

$$A = \begin{cases} 0, & \text{Bottom layer nodes} \\ -Fo, & \text{Top layer nodes} \end{cases}$$

$$B = 1 + Fo \left[ 1 + c_3(h_{c_e} + h_{r_e}) + c_4(h_{c_o} + h_{r_o}) \right]$$

$$C = \begin{cases} -Fo, & \text{Top layer nodes} \\ 0, & \text{Bottom layer nodes} \end{cases}$$

$$\begin{aligned} D = T^{\wedge} + Fo & \left[ c_1(T_E^{\wedge} + T_W^{\wedge} - 2T^{\wedge}) + c_2(T_I^{\wedge} - T^{\wedge}) \right. \\ & + c_3(q''_e + h_{c_e}T_{\infty} + h_{r_e}T_{sur}) \\ & \left. + c_4(q''_o - h_{c_o}T_{\infty} - h_{r_o}T_{sur}) \right] \end{aligned}$$

### 2. Step 2 (φ-Direction)

$$AT_E^{**} + BT^{**} + CT_W^{**} = D$$

where

$$A = -c_1 Fo$$

$$B = 1 + Fo \left[ 2c_1 + c_3(h_{c_e} + h_{r_e}) + c_4(h_{c_o} + h_{r_o}) \right]$$

$$C = -c_1 Fo$$

$$\begin{aligned} D = T^* + Fo & \left[ (T_e^* - T^*) + c_2(T_i^* - T^*) \right. \\ & + c_3(q''_e + h_{c_e}T_\infty + h_{r_e}T_{sur}) \\ & \left. + c_4(q''_o + h_{c_o}T_\infty + h_{r_o}T_{sur}) \right] \end{aligned}$$

### 3. Step 3 (r-Direction)

$$AT_i^{***} + BT^{***} + CT_o^{***} = D$$

where

$$A = -c_2 Fo$$

$$B = 1 + Fo \left[ c_2 + c_3(h_{c_e} + h_{r_e}) + c_4(h_{c_o} + h_{r_o}) \right]$$

$$C = 0$$

$$\begin{aligned} D = T^* + Fo & \left[ (T_e^* - T^*) + c_1(T_E^{**} + T_W^{**} - 2T^{**}) \right. \\ & \left. + c_3(q''_e + h_{c_e}T_\infty + h_{r_e}T_{sur}) \right] \end{aligned}$$

$$+ c_4(q''_o + h_{c_o} T_{\infty} + h_{r_o} T_{sur}) \}$$

#### 4. Step 4 (New Temperatures)

$$T^{n+1} = \frac{1}{c_5} \{ T^n + 2Fo \left[ (T_s^* - T^*) + c_1(T_E^{**} + T_W^{**} - 2T^{**}) \right.$$

$$+ c_2(T_I^{***} - T^{***})$$

$$+ c_3(q''_s + h_{c_s} T_{\infty} + h_{r_s} T_{sur})$$

$$\left. + c_4(q''_o + h_{c_o} T_{\infty} + h_{r_o} T_{sur}) \right\}$$

#### D. TYPE 4 NODES—MID-LAYERS, INNER SURFACE

$$c_1 = \frac{k_z \Delta z^2}{k_z r(r + \frac{\Delta r}{4}) \Delta \phi^2}$$

$$c_2 = \frac{2 k_r (r + \frac{\Delta r}{2}) \Delta z^2}{k_z (r + \frac{\Delta r}{4}) \Delta r^2}$$

$$c_3 = \frac{2 r \Delta z^2}{k_z (r + \frac{\Delta r}{4}) \Delta r}$$

$$c_6 = 1 + Fo [c_3(h_{c_i} + h_{r_i})]$$

$$h_{r_i} = \epsilon_i \sigma (T^n + T_{sur}) [(T^n)^2 + T_{sur}^2]$$

### 1. Step 1 (z-Direction)

$$AT_s^* + BT^* + CT_N^* = D$$

where

$$A = -\frac{Fo}{2}$$

$$B = 1 + \frac{Fo}{2} [2 + c_3(h_{e_i} + h_{r_i})]$$

$$C = -\frac{Fo}{2}$$

$$D = T^* + \frac{Fo}{2} [c_1(T_E^* + T_w^* - 2T^*) + c_2(T_o^* - T^*) \\ + c_3(q''_i + h_{e_i}T_\infty + h_{r_i}T_{sur})]$$

### 2. Step 2 ( $\phi$ -direction)

$$AT_E^{**} + BT^{**} + CT_w^{**} = D$$

where

$$A = -\frac{c_1 Fo}{2}$$

$$B = 1 + \frac{Fo}{2} [2c_1 + c_3(h_{e_i} + h_{r_i})]$$



$$C = -\frac{c_1 Fo}{2}$$

$$D = T^* + \frac{Fo}{2} \left[ (T_N^* + T_S^* - 2T^*) + c_2 (T_o^* - T^*) \right. \\ \left. + c_3 (q''_i + h_{c,i} T_\infty + h_{r,i} T_{sur}) \right]$$

### 3. Step 3 (r-Direction)

$$AT_i^{***} + BT^{***} + CT_o^{***} = D$$

where

$$A = 0$$

$$B = 1 + \frac{Fo}{2} \left[ c_2 + c_3 (h_{c,i} + h_{r,i}) \right]$$

$$C = -\frac{c_2 Fo}{2}$$

$$D = T + \frac{Fo}{2} \left[ (T_N^* + T_S^* - 2T^*) + c_1 (T_E^{**} + T_W^{**} - 2T^{**}) \right. \\ \left. + c_3 (q''_i + h_{c,i} T_\infty + h_{r,i} T_{sur}) \right]$$

### 4. Step 4 (New Temperatures)

$$T^{n+1} = \frac{1}{c_6} \left\{ T^* + Fo \left[ (T_N^* + T_S^* - 2T^*) + c_1 (T_E^{**} + T_W^{**} - 2T^{**}) \right. \right. \\ \left. \left. + c_3 (q''_i + h_{c,i} T_\infty + h_{r,i} T_{sur}) \right] \right\}$$

$$+ c_2(T_o^{***} - T^{***})$$

$$+ c_3(q''_1 + h_{c_1}T_{\infty} + h_{r_1}T_{sur}) \Big] \Big\}$$

#### E. TYPE 4 NODES—MID-LAYERS, MID-RADI

$$c_1 = \frac{k_\phi \Delta z^2}{k_r r^2 \Delta \phi^2}$$

$$c_2 = \frac{k_r (r + \frac{\Delta r}{2}) \Delta z^2}{k_r r \Delta r^2}$$

$$c_3 = \frac{(r + \frac{\Delta r}{2}) \Delta z^2}{k_r \Delta r^2}$$

##### 1. Step 1 (z-Direction)

$$AT_s^* + BT^* + CT_N^* = D$$

where

$$A = -\frac{Fo}{2}$$

$$B = 1 + Fo$$

$$C = -\frac{Fo}{2}$$

$$D = T + \frac{Fo}{2} [c_1(T_E^* + T_w^* - 2T^*) + c_2(T_I^* - T^*) + c_3(T_o^* - T^*)]$$

## 2. Step 2 ( $\phi$ -Direction)

$$AT_E^{**} + BT^{**} + CT_w^{**} = D$$

where

$$A = -\frac{c_1 Fo}{2}$$

$$B = 1 + c_1 Fo$$

$$C = -\frac{c_1 Fo}{2}$$

$$D = T + \frac{Fo}{2} [(T_N^* + T_s^* - 2T^*) + c_2(T_I^* - T^*) + c_3(T_o^* - T^*)]$$

## 3. Step 3 (r-Direction)

$$AT_I^{***} + BT^{***} + CT_o^{***} = D$$

where

$$A = -\frac{c_2 Fo}{2}$$

$$B = 1 + \frac{Fo}{2} (c_2 + c_3)$$

$$C = -\frac{c_3 Fo}{2}$$

$$D = T^* + \frac{Fo}{2} \left[ (T_N^* + T_S^* - 2T^*) + c_1 (T_E^{**} + T_W^{**} - 2T^{**}) \right]$$

#### 4. Step 4 (New Temperatures)

$$T^{n+1} = T^* + Fo \left[ (T_N^* + T_S^* - 2T^*) + c_1 (T_E^{**} + T_W^{**} - 2T^{**}) + c_2 (T_I^{***} - T^{***}) + c_3 (T_O^{***} - T^{***}) \right]$$

#### F. TYPE 4 NODES—MID-LAYERS, OUTER SURFACE

$$c_1 = \frac{k_\phi \Delta z^2}{k_z r (r - \frac{\Delta r}{4}) \Delta \phi^2}$$

$$c_2 = \frac{2k_r (r - \frac{\Delta r}{4}) \Delta z^2}{k_z (r - \frac{\Delta r}{4}) \Delta r^2}$$

$$c_3 = \frac{2r \Delta z^2}{k_z (r - \frac{\Delta r}{4}) \Delta r}$$

$$c_4 = 1 + Fo \left[ c_3 (h_{e_o} + h_{r_o}) \right]$$

$$h_{r_o} = \epsilon_o \sigma (T^* + T_{sur}) \left[ (T^*)^2 + T_{sur}^2 \right]$$

### 1. Step 1 (z-Direction)

$$AT_s^* + BT^* + CT_N^* = D$$

where

$$A = -\frac{Fo}{2}$$

$$B = 1 + \frac{Fo}{2} \left[ 2 + c_3(h_{c_o} + h_{r_o}) \right]$$

$$C = -\frac{Fo}{2}$$

$$+ c_3(q''_o + h_{c_o}T_{\infty} + h_{r_o}T_{sur}) \Big]$$

### 2. Step 2 ( $\phi$ -direction)

$$AT_E^{**} + BT^{**} + CT_w^{**} = D$$

where

$$A = -\frac{c_1 Fo}{2}$$

$$B = 1 + \frac{Fo}{2} + 2c_1 + c_3(h_{c_o} + h_{r_o})$$

$$C = -\frac{c_1 Fo}{2}$$

$$D = T^* + \frac{Fo}{2} \left[ (T_N^* + T_s^* - 2T^*) + c_2(T_I^* - T^*) \right. \\ \left. + c_3(q''_o + h_{c_o}T_\infty + h_{r_o}T_{sur}) \right]$$

### 3. Step 3 (r-Direction)

$$AT_I^{***} + BT_o^{***} + CT_o^{***} = D$$

where

$$A = -\frac{c_2 Fo}{2}$$

$$B = 1 + \frac{Fo}{2} c_2 + c_3(h_{c_o} + h_{r_o})$$

$$C \approx 0$$

$$D = T^* + \frac{Fo}{2} \left[ (T_N^* + T_s^* - 2T^*) + c_1(T_E^* + T_w^* - 2T^*) \right. \\ \left. + c_3(q''_o + h_{c_o}T_\infty + h_{r_o}T_{sur}) \right]$$

### 4. Step 4 (New Temperatures)

$$T^{*+1} = \frac{1}{c_4} \left\{ T^* + Fo \left[ (T_N^* + T_s^* - 2T^*) + c_1(T_E^* + T_w^* - 2T^*) \right. \right. \\ \left. \left. + c_2(T_I^{***} - T^{***}) \right. \right. \\ \left. \left. + c_3(q''_o + h_{c_o}T_\infty + h_{r_o}T_{sur}) \right] \right\}$$

APPENDIX C  
VALIDATION MODEL

**A PROBLEM SPECIFICATION**

Given a right circular cylinder with adiabatic ends and specified initial temperature distribution and constant inner-surface temperature of zero degrees Celsius. At time  $t = 0$ , a uniform constant heat flux is applied to the outer surface of the cylinder. Assume constant properties and no losses due to convection or radiation.

**B ANALYTIC SOLUTION**

**Fourier Equation**

$$\nabla^2 T = \frac{1}{\alpha} \frac{\partial T}{\partial t}$$

**Boundary Conditions:**

$$r_o = a \qquad T = 0$$

$$r_o = b \qquad \frac{\partial T}{\partial t} = \frac{q''}{k}$$

**Initial Condition:**

$$t = 0 \qquad T = 0$$

Let:  $T = \Psi(r_o, t) + \Phi(r_o)$

### Auxiliary Problem

$$\nabla^2 \Psi(r_o, t) + \nabla^2 \Phi(r_o) = \frac{1}{\alpha} \frac{\partial \Psi(r_o, t)}{\partial t}$$

$$r_o = a \quad \Psi(r_o, t) + \nabla^2 \Phi(r_o) = \frac{1}{\alpha} \frac{\partial \Psi}{\partial t}$$

$$r_o = b \quad \frac{\partial \Psi}{\partial r}(r_o, t) + \frac{d\Phi}{dr_o} = \frac{q''}{k}$$

$$t = 0 \quad \Psi(r_o, 0) + \Phi(r_o) = 0$$

### Separating the Auxiliary Problem

Equation 1:

$$\frac{d^2 \Phi}{dr_o^2} + \frac{1}{r_o} \frac{d\Phi}{dr_o} = 0$$

$$r_o = a \quad \Phi(a) = 0$$

$$r_o = b \quad \frac{d\Phi}{dr_o} = \frac{q''}{k}$$

Let:  $y = \frac{d\Phi}{dr_o}$

Then:

$$\frac{dy}{dr_o} = -\frac{y}{r_o}$$



$$\frac{dy}{y} = - \frac{dr_o}{r_o}$$

$$\ln(y) = - \ln(r_o) + c_1$$

$$y = \frac{c_1}{r_o}$$

And:

$$\frac{d\Phi}{dr_o} = \frac{c_1}{r_o}$$

$$d\Phi = c_1 \frac{dr_o}{r_o}$$

Yielding:

$$\Phi = c_1 \ln(r_o) + c_2$$

### Evaluating Constants

$$r = a \quad \rightarrow \quad \Phi = 0 \quad \Rightarrow \quad 0 = c_1 \ln(a) + c_2$$

$$c_2 = - \frac{b q''}{k} \ln(a)$$

$$r = b \quad \rightarrow \quad \frac{d\Phi}{dr_o} = \frac{q''}{k} \quad \Rightarrow \quad c_1 = \frac{b \cdot q''}{k}$$

$$\text{Thus: } \Phi(r_o) = \frac{b q''}{k} \ln\left(\frac{r_o}{a}\right)$$

### Transforming the Variable

$$\text{Let: } r = \frac{r_o}{b}$$

$$\text{And: } \kappa = \frac{a}{b}$$

$$\text{Then: } \Phi(r) = \frac{b q''}{k} \ln\left(\frac{r}{\kappa}\right)$$

### Converting Boundary Conditions

$$r_o = a \quad \Rightarrow \quad r = \frac{r_o}{b} = \kappa$$

$$r_o = b \quad \Rightarrow \quad r = 1$$

The Transformed Problem Becomes

$$\Phi(r) = \frac{b q''}{k} \ln\left(\frac{r}{\kappa}\right)$$

$$r = \kappa \quad \phi = 0$$

$$r = 1 \quad \frac{d\phi(r)}{dr} = \frac{b q''}{k}$$

**Equation 2:**

$$\frac{\partial^2 \Psi}{\partial r_o^2} + \frac{1}{r_o} \frac{\partial \Psi}{\partial r_o} = \frac{1}{\alpha} \frac{\partial \Psi}{\partial t}$$

$$r_o = a \quad \Psi(a) = 0$$

$$r_o = b \quad \frac{\partial \Psi}{\partial r_o} = 0$$

$$t = 0 \quad \Psi(r_o, 0) = -\Phi(r_o)$$

$$\text{Let: } \Psi(r_o, t) = R(r_o) \Gamma(t)$$

$$\text{Then: } R'' \Gamma + \frac{1}{r_o} R' \Gamma = \frac{1}{\alpha} R \Gamma'$$

Which reduces to

$$\frac{R''}{R} + \frac{1}{r_o} \frac{R'}{R} = \frac{1}{\alpha} \frac{\Gamma'}{\Gamma} = -\beta^2$$

Separating the two equalities, we obtain

**Equation 2.a**

$$\frac{1}{\alpha} \cdot \frac{\Gamma'}{\Gamma} = -\beta^2$$

$$\ln(\Gamma) = -\alpha \beta^2 t$$

$$\Gamma(t) = e^{-\alpha \beta^2 t}$$

**Equation 2.b**

$$r_o = a \quad R = 0$$

$$r_o = b \quad \frac{dR}{dr_o} = 0$$

From Table 3-2, Case B of Özisik

$$R(\beta^*, r_0) = J_0(\beta^* r_0) \cdot Y'_0(\beta^* b) - J'_0(\beta^* b) \cdot Y_0(\beta^* r_0)$$

the inverse of the norm is given by

$$\frac{1}{N(\beta^*)} = \frac{\frac{\pi^2}{2} \beta^{*2} J_0^2(\beta^* a)}{J_0^2(\beta^* a) - J_0^2(\beta^* b)}$$

and the eigenvalues are the roots of

$$J_0(\beta^* a) \cdot Y'_0(\beta^* b) - J'_0(\beta^* b) \cdot Y_0(\beta^* a) = 0$$

From the recursion relations we know

$$J'_0(x) = -J_1(x)$$

and similarly

$$Y'_0(x) = -Y_1(x)$$

Substituting, multiplying by -1, and rearranging terms yields

$$Y_1(\beta^* b) \cdot J_0(\beta^* a) - J_1(\beta^* b) \cdot Y_0(\beta^* a) = 0$$

### Transforming Variables:

Letting:  $\beta = \beta^* b$

and recalling that:  $\kappa = \frac{a}{b}$

we obtain by substitution the common tabular form:

$$Y_1(\beta) \cdot J_0(\kappa \beta) - J_1(\beta) \cdot Y_0(\kappa \beta) = 0$$

Performing similar variable transformations on the transient solution yields:

$$R(\beta, r) = J_1(\beta) \cdot Y(r\beta) - J_0(r\beta) \cdot Y_1(\beta)$$

with the norm

$$N(\beta) = \frac{2}{\pi^2} \frac{J_0^2(\kappa\beta) - J_1^2(\beta)}{\frac{\beta^2}{b^2} J_0^2(\kappa\beta)}$$

**Total Solution:**

$$T(r, t) = \frac{bq}{k} \ln\left(\frac{r}{k}\right) \sum_i \left\{ \exp\left(-\frac{\alpha \beta_i^2 t}{b^2}\right) \frac{1}{N_i(\beta)} R_i(r, \beta) \right.$$

$$\left. \cdot \int_{\kappa}^1 \left[ \rho R_i(r, \beta) \frac{b \cdot q}{\kappa} \ln\left(\frac{\rho}{\kappa}\right) \right] d\rho \right\}$$

## APPENDIX D

### EXPLICIT METHOD STABILITY REQUIREMENTS

The following equations represent the limiting time step size for their respective node types.

#### A TYPE 1 NODES—END LAYER, INNER SURFACE

$$\Delta t_1 \leq 1 + \frac{k_\phi \Delta z^2}{k_z r (r + \frac{\Delta r}{4}) \Delta \phi^2} + \frac{k_r (r + \frac{\Delta r}{2}) \Delta z^2}{k_z (r + \frac{\Delta r}{4}) \Delta r^2}$$

#### B TYPE 2 NODES—END LAYERS, MID-RADII

$$\Delta t_2 \leq 1 + \frac{k_\phi \Delta z^2}{k_z r^2 \Delta \phi^2} + \frac{k_r (r - \frac{\Delta r}{2}) \Delta z^2}{2 k_z r \Delta r^2}$$

#### C TYPE 3 NODES—END LAYERS, OUTER SURFACE

$$\Delta t_3 \leq 1 + \frac{k_\phi \Delta z^2}{k_z r (r + \frac{\Delta r}{4}) \Delta \phi^2} + \frac{k_r (r + \frac{\Delta r}{2}) \Delta z^2}{k_z (r + \frac{\Delta r}{4}) \Delta r^2}$$

#### D TYPE 4 NODES—MID-LAYERS, INNER SURFACE

$$\Delta t_4 \leq 2 + \frac{2 k_\phi \Delta z^2}{k_z (r - \frac{\Delta r}{4}) \Delta \phi^2} + \frac{2 k_r (r + \frac{\Delta r}{4})}{\Delta r^2}$$

**E. TYPE 5 NODES—MID-LAYERS, MID-RADI**

$$\Delta t_5 \leq 2 + \frac{2k_\phi \Delta z^2}{k_z r^2 \Delta \phi^2} + \frac{k_r(r - \frac{\Delta r}{2})}{k_z r \Delta r^2} + \frac{k_r(r + \frac{\Delta r}{2})}{k_z r \Delta r^2}$$

**F. TYPE 6 NODES—MID-LAYERS, OUTER-SURFACE**

$$\Delta t_6 \leq \frac{2k_\phi \Delta z^2}{k_z r(r - \frac{\Delta r}{4}) \Delta \phi^2} + \frac{2k_r(r - \frac{\Delta r}{2}) \Delta z^2}{k_z(r - \frac{\Delta r}{4}) \Delta r^2}$$

## APPENDIX E

### EXPLICIT METHOD CODE

```

C      ECR FORTRAN A - (REV. 11/23/88)
C      SATELLITE MODEL - VALIDATED 11/26/88
C      *****
C      * ECR - PROGRAM TO COMPUTE NODE TEMPERATURES FOR A RIGHT *
C      *      CIRCULAR CYLINDER SUBJECTED TO CONVECTIVE AND/OR *
C      *      RADIATIVE BOUNDARY CONDITIONS USING THE EXPLICIT *
C      *      METHOD OF FINITE DIFFERENCE ANALYSIS.          *
C      *****
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C      DIMENSION TOLD(15,15,15),TNEW(15,15,15)
C      DIMENSION DMAT(15,15,15)

C      COMMON /ARRSIZ/  III,JJJ,KKK
C      COMMON /CONDUCT/ DKR,DKP,DKZ
C      COMMON /CONVEC/  HSUBN,HSUBS,HSUBI,HSUBO
C      COMMON /GEOMET/  DPHI,DR,DZ,RAD,RMAX,RMIN,THICK,ZMAX,LPHI
C      COMMON /HEAT/    ALFA,FO,QPSUBN,QPSUBS,QPSUBI,QPSUBO
C      COMMON /MATSIZ/  NODES,NPHI,NLAYER
C      COMMON /PARAMS/  CSUBP,RHO
C      COMMON /RADN/    EPSUBN,EPSUBS,EPSUBI,EPSUBO,SIGMA
C      COMMON /TEMPS/   T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,TINF,TSUR
C      COMMON /TIMES/   DELTIM,DTMAX,FINTIM,LIMIT

C      START TIMER
C      CALL STIME
C      OPEN (UNIT=11,FILE='ECRLNG')
C      OPEN (UNIT=14,FILE='ECRDAT')
C      OPEN (UNIT=15,FILE='ECRCPU')
C      ***** USER DEFINED VARIABLES *****
C      DELTIM = 0.30D+00
C      FINTIM = 3000.0D+00
C      IPRINT = 100
C      ACTUAL MATRIX DIMENSIONS
C      NODES = 10
C      NPHI = 4
C      NLAYER = 5
C      NODE SPECIFICATION FOR FILE ESHORT
C      MRAD = 10
C      MPHI = 2
C      MLAYER = 3
C      DECLARED ARRAY DIMENSIONS
C      III = 15
C      JJJ = 15
C      KKK = 15
C      ABSORPTIVITY COEFFICIENTS
C      ALFAN = 0.4D+00
C      ALFAS = 0.4D+00

```



```

      ALFAO = 0.4D+00
C     ALBEDO COEFFICIENT
      ALBEDO = 0.30D+00
C     THERMAL CONDUCTIVITIES
      DKR = 177.D+00
      DKP = 177.D+00
      DKZ = 177.D+00
C     EMISSIVITIES
      EPSUBN = 0.9D+00
      EPSUBS = 0.9D+00
      EPSUBI = 0.D+00
      EPSUBO = 0.9D+00
C     CONVECTION COEFFICIENTS
      HSUBN = 0.D+00
      HSUBS = 0.D+00
      HSUBI = 0.D+00
      HSUBO = 0.D+00
C     HEAT FLUXES
      SOLAR = 1353.D+00
      EIR   = 238.0D+00
      GEN   = 1000.
      QPSUBI = 0.D+00
      QPSUBO = 0.D+00
      QPSUBN = 0.D+00
      QPSUBS = 0.D+00
C     GEOMETRY
      RMIN = 0.15244D+00
      RMAX = 0.22866D+00
      ZMAX = 1.D+00
C     VIEW FACTORS WITH THE EARTH
      VERTHN = 0.D+00
      VERTHS = 1.D+00
      VERTHI = 0.D+00
      VERTH0 = 0.D+00
C     VIEW FACTORS WITH SPACE
      VSPCN = 1.D+00
      VSPCS = 0.D+00
      VSPCI = 0.D+00
      VSPCO = 1.D+00
C     TEMPERATURES
      TABS = 273.D+00
      TINIT = 293.D+00
      TINF  = 0.D+00
      TSUR  = 4.D+00
C     MATERIAL PROPERTIES
      CSUBP = 875.0D+00
      RHO   = 2270.0D+00
C     INITIALIZE OTHER VARIABLES
      ICOUNT = 0
      ITIME  = 0
      ORBTIM = 0.D+00
      PI     = 4.0D+00*DTAN(1.D+00)
      RAD    = RMIN
      SIGMA  = 5.67D-08
      TIME   = 0.D+00

```

```

      T1      = 0.D+00
C      INITIALIZE MATRICES
      CALL BMINIT (TOLD,TINIT)
      CALL BMINIT (TNEW,TINIT)
C      SURFACE AREA CALCULATIONS
      SAREAN = PI*(RMAX**2 - RMIN**2)
      SAREAS = PI*(RMAX**2 - RMIN**2)
      SAREAI = PI*(2*RMIN)*ZMAX
      SAREAO = PI*(2*RMAX)*ZMAX
C      INITIAL CALCULATIONS
      ALFA = DKZ/(RHO*CSUBP)
      DPHI = 2.*PI/NPHI
      DR = (RMAX-RMIN)/(NODES-1)
      DZ = ZMAX/(NLAYER-1)
      LPHI = 360/NPHI
      QPSUBI = GEN/SAREAI
C      ORBITAL PARAMETERS
      PERIOD = 5440.D+00
      ECLIPS = 2181.9D+00
      OMEGA = 1.155D-03
      TAU = 0.01*PERIOD/4
      TIME1 = 0.D+00
      TIME2 = PERIOD/4.D+00
      TIME3 = 107.8D+00/360 * PERIOD
      TIME4 = (270 - 17.8)/360 * PERIOD - 4*TAU
      TIME5 = (270 - 17.8)/360 * PERIOD
      TIME6 = 270/360 * PERIOD
C      ESTABLISH STABILITY REQUIREMENT
      CALL STABIL (DTMAX)
      FO = ALFA*DELTIM/(DZ**2)
      LIMIT = FINTIM/DELTIM + 500
C      WRITE MATRIX DIMENSIONS TO FILE 14 (ECRDAT)
      WRITE (14,*) NODES,NPHI,NLAYER
C
C      *** PRINT OPTION 3 - SAVES DESIGNATED NODE TEMPS AT SPECIFIED
C      PRINT INTERVAL.  USE FOR LONG PERIOD EVALS.
      WRITE (11,*) 'ECRSAT'
      WRITE (11,*)
      WRITE (11,888)
888  FORMAT (T6,'TIME',T18,'DELTIM',T30,'LAYER1',T45,'LAYER3',T60,'LAYER
      1R')
      WRITE (11,*)
C
C      CMMMMMMMMMMMMMMMMMMMM BEGIN TEMPERATURE COMPUTATIONS MMMMMMMMMMMMMMMMMMMMM
C      GET TIME ENTERING LOOP (MAINFRAME VERSIONS OF CODE ONLY)
      WRITE (11,*) 'ENTERING LOOP TIME'
      CALL GTIME(1,11)
      DO 100 I=1,LIMIT
          ICOUNT=ICOUNT+1
          TIME=TIME+DELTIM
          ORBTIM = ORBTIM + DELTIM
          IF (ORBTIM.GE.PERIOD) ORBTIM = ORBTIM - PERIOD
C      SURFACE FLUX CALCULATIONS (TIME DEPENDENT)
      IF ((ORBTIM.GE.0).AND.(ORBTIM.LE.TIME2)) THEN
          QPSUBN = ALFAN*SOLAR*DCOS(OMEGA*ORBTIM)

```

```

      QPSUBS = ALFAS*(EIR + ALBEDO*SOLAR*DCOS(OMEGA*ORBTIM))
      QPSUBO = 0.5*ALFAO*SOLAR*DSIN(OMEGA*ORBTIM)
    ELSEIF ((ORBTIM.GT.TIME2).AND.(ORBTIM.LE.TIME3)) THEN
      QPSUBN = 0.
      QPSUBS = ALFAS*(EIR + (ALBEDO+DCOS(OMEGA*ORBTIM))*SOLAR)
      QPSUBO = 0.5*ALFAO*SOLAR*DSIN(OMEGA*ORBTIM)
    ELSEIF ((ORBTIM.GT.TIME3).AND.(ORBTIM.LE.TIME4)) THEN
      QPSUBN = 0.
      QPSUBS = ALFAS*EIR
      QPSUBO = 0.D+00
    ELSEIF ((ORBTIM.GT.TIME4).AND.(ORBTIM.LE.TIME5)) THEN
      QPSUBN = 0.D+00
      QPSUBS = ALFAS*(EIR
1          + DABS(ALBEDO*SOLAR*DSIN(OMEGA*ORBTIM))
1          * (1-DEXP(-(ORBTIM-TIME4)/TAU)))
      QPSUBO = 0.5*ALFAO*DABS(ALBEDO*SOLAR
1          + DSIN(OMEGA*ORBTIM)
1          * (1-DEXP(-(ORBTIM-TIME4)/TAU)))
    ELSEIF ((ORBTIM.GT.TIME5).AND.(ORBTIM.LT.TIME6)) THEN
      QPSUBN = 0.D+00
      QPSUBS = ALFAS*(EIR
1          + DABS(ALBEDO*SOLAR*DSIN(OMEGA*ORBTIM)))
      QPSUBO = 0.5*ALFAO*DABS(SOLAR*DSIN(OMEGA*ORBTIM))
    ELSEIF (ORBTIM.GT.TIME6) THEN
      QPSUBN = ALFAN*DABS(SOLAR*DCOS(OMEGA*ORBTIM))
      QPSUBS = ALFAS*(EIR
1          + DABS(ALBEDO*SOLAR*DSIN(OMEGA*ORBTIM)))
      QPSUBO = 0.5*ALFAO*DABS(SOLAR*DSIN(OMEGA*ORBTIM))
    ENDIF
C   CALL MAIN SUBROUTINE
      CALL NUTEMP (TOLD,TNEW,DELTIM)
C   MAP NEW TEMPERATURE MATRIX ONTO OLD TEMPERATURE MATRIX
      CALL MAP122 (TNEW,TOLD)
C   SAVE TIME, DELTIM AND TEMPERATURE MATRIX TO DATA FILE 14 (ECRDAT)
      IF (ICOUNT.EQ.IPRINT) THEN
        ICOUNT = 0
        WRITE (14,*) TIME,DELTIM
C
C   **** CONVERT NEW TEMPERATURES TO EXCESS FOR PRINTOUT ****
      DO 200 LAYER=1,NLAYER
        DO 300 INCRAD=1,NODES
          DO 400 IPHI=1,NPHI
            TNEW(INCRAD,IPHI,LAYER) = TNEW(INCRAD,IPHI,LAYER 1) - TINIT
C
C   **** PRINT OPTION 1 - USE WITH CMPSHT,COMPAR,LNGFRM,SHTFRM,BCRERR
C   SHORT EVALUATION TIMES ONLY (<400 SEC)
C   WRITE (14,*) TNEW(INCRAD,IPHI,LAYER)
C   CONTINUE
400    CONTINUE
300    CONTINUE
200    CONTINUE
C
C   *** PRINT OPTION 2 - SAVES COMPLETE MATRIX AT SPECIFIED PRINT
C   INTERVAL - SWITCH OFF ALL SUBROUTINES LISTED
C   UNDER PRINT OPTION 1 FIRST.
C   CALL PRMAT (TIME,DTIME,TNEW,NODES,NPHI,NLAYER)

```



CS

```

C
C      STABIL FORTRAN A (REV. 11/14/88)
C      *****
C      * STABIL - SUBROUTINE TO COMPUTE THEORETICAL MAXIMUM TIME *
C      *      INCREMENT ALLOWED TO MAINTAIN STABILITY IN EXPLICIT *
C      *      FINITE DIFFERENCE ANALYSIS OF A CYLINDER IN 3-D *
C      *      POLAR CYLINDRICAL COORDINATES. *
C      *****
C      PROGRAM LINE 237
C      SUBROUTINE STABIL (DTMAX)
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C      DIMENSION DT(6)
C      COMMON /CONVEC/ DKR,DKP,DKZ
C      COMMON /CONVEC/ HSUBN,HSUBS,HSUBI,HSUBO
C      COMMON /GEOMET/ DPHI,DR,DZ,RAD,RMAX,RMIN,THICK,ZMAX,LPHI
C      COMMON /HEAT/ ALFA,FO,QPSUBN,QPSUBS,QPSUBI,QPSUBO
C      COMMON /RADN/ EPSUBN,EPSUBS,EPSUBI,EPSUBO,SIGMA
C      *** BODY ***
C      DO 50 I=1,6
C          DT(I) = 0.D+00
50      CONTINUE
C      DUMMY = (DZ**2)/ALFA
C      END LAYERS, INNER SURFACE NODES
C          DCREI = 1 + (2*DKP*DZ**2)/(2*DKZ*RAD*(RAD+DR/4)*DPHI**2)
1+      (DKR*(RAD+DR/2)*DZ**2)/(DKZ*(RAD+DR/4)*DR**2)
C          DT(1) = DUMMY/(2*DCREI)
C          DTMAX = DT(1)
C      END LAYERS, MID-RADII NODES
C          DCREM = 1 + 2*(DKP*DZ**2)/(2*DKZ*RAD**2*DPHI**2)
1+      (DKR*(RAD-DR/2)*DZ**2)/(2*DKZ*RAD*DR**2)
2+      (DKR*(RAD+DR/2)*DZ**2)/(2*DKZ*RAD*DR**2)
C          DT(2) = DUMMY/(2*DCREM)
C      END LAYERS, OUTER SURFACE NODES
C          DCREO = 1 + 2*(DKP*DZ**2)/(2*DKZ*(RAD-DR/4)*RAD*DPHI**2)
1+      (DKR*(RAD-DR/2)*DZ**2)/(DKZ*(RAD-DR/4)*DR**2)
C          DT(3) = DUMMY/(2*DCREO)
C      MID-LAYERS, INNER SURFACE NODES
C          DCRMI = 2 + 2*(DKP*DZ**2)/(DKZ*RAD*(RAD+DR/4)*DPHI**2)
1+      (2*DKR*(RAD+DR/2)*DZ**2)/(DKZ*(RAD+DR/4)*DR**2)
C          DT(4) = DUMMY/DCRMI
C      MID-LAYERS, MID-RADII NODES
C          DCRMM = 2 + 2*(DKP*DZ**2)/(DKZ*RAD**2*DPHI**2)
1+      (DKR*(RAD-DR/2)*DZ**2)/(DKZ*RAD*DR**2)
2+      (DKR*(RAD+DR/2)*DZ**2)/(DKZ*RAD*DR**2)
C          DT(5) = DUMMY/DCRMM
C      MID-LAYERS, OUTER SURFACE NODES
C          DCRMO = 2 + 2*(DKP*DZ**2)/(DKZ*RAD*(RAD-DR/4)*DPHI**2)
1+      (2*DKR*(RAD-DR/2)*DZ**2)/(DKZ*(RAD-DR/4)*DR**2)
C          DT(6) = DUMMY/DCRMO
C      DO 100 I=1,6
C          IF ((DT(I).GT.0).AND.(DTMAX.GT.DT(I))) DTMAX = DT(I)
C          WRITE (*,*) 'FROM STABIL, I,DT(I)',I,DT(I)
100      CONTINUE
C      IF (DTMAX.LE.0) WRITE (*,*) 'STABIL - DTMAX LE ZERO'

```

```

998      RETURN
        END
C
C
C*****
C
C      NUTEMP FORTRAN A - (REV. 11/14/88)
C      *****
C      * NUTEMP - THIS SUBROUTINE COMPUTES THE MATRIX OF NODE *
C      *          TEMPERATURES AT THE N+1 TIMESTEP FOR THE   *
C      *          EXPLICIT METHOD OF FINITE DIFFERENCE         *
C      *          ANALYSIS FOR A RIGHT CIRCULAR CYLINDER WITH *
C      *          CONSTANT SURFACE HEAT FLUX AND A SPECIFIED  *
C      *          INNER SURFACE TEMPERATURE.                  *
C      *****
C      PROGRAM LINE 298
C      SUBROUTINE NUTEMP (TOLD,TNEW,DELTIM)
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C      COMMON /ARRSIZ/ III,JJJ,KKK
C      COMMON /CONDUCT/ DKR,DKP,DKZ
C      COMMON /CONVEC/ HSUBN,HSUBS,HSUBI,HSUBO
C      COMMON /GEOMET/ DPHI,DR,DZ,RAD,RMAX,RMIN,THICK,ZMAX,LPHI
C      COMMON /HEAT/ ALFA,FO,QPSUBN,QPSUBS,QPSUBI,QPSUBO
C      COMMON /MATSIZ/ NODES,NPHI,NLAYER
C      COMMON /NPOSIT/ INCRAD,IPHI,LAYER
C      COMMON /PARAMS/ CSUBP,RHO
C      COMMON /RADN/ EPSUBN,EPSUBS,EPSUBI,EPSUBO,SIGMA
C      COMMON /TEMPS/ T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,TINF,TSUR
C      DIMENSION TOLD(III,JJJ,KKK),TNEW(III,JJJ,KKK)
C      CB***** BOTTOM LAYER *****
C      LAYER = 1
C      **** BOTTOM LAYER, INNER RADIUS ****
C      INCRAD = 1
C      RAD = RMIN
C      ***** BOTTOM LAYER, INNER RADIUS, FIRST PHI *****
C      IPHI =1
C      T1=TOLD(INCRAD,IPHI,LAYER)
C      T2=TOLD(INCRAD,IPHI,(LAYER+1))
C      T3=TOLD(INCRAD,NPHI,LAYER)
C      T4=TOLD(INCRAD,(IPHI+1),LAYER)
C      T5=TOLD((INCRAD+1),IPHI,LAYER)
C      CALL ECREI (TEMP)
C      TNEW(INCRAD,IPHI,LAYER)=TEMP
C      ***** BOTTOM LAYER, INNER RADIUS, MID-PHIS *****
C      DO 100 IPHI=2,(NPHI-1)
C      T1=TOLD(INCRAD,IPHI,LAYER)
C      T2=TOLD(INCRAD,IPHI,(LAYER+1))
C      T3=TOLD(INCRAD,(IPHI-1),LAYER)
C      T4=TOLD(INCRAD,(IPHI+1),LAYER)
C      T5=TOLD((INCRAD+1),IPHI,LAYER)
C      CALL ECREI (TEMP)
C      TNEW(INCRAD,IPHI,LAYER)=TEMP
100     CONTINUE
C      ***** BOTTOM LAYER, INNER RADIUS, LAST PHI *****
C      IPHI=NPHI

```

```

      T1=TOLD (INCRAD, IPHI, LAYER)
      T2=TOLD (INCRAD, IPHI, (LAYER+1))
      T3=TOLD (INCRAD, (IPHI-1), LAYER)
      T4=TOLD (INCRAD, 1, LAYER)
      T5=TOLD ((INCRAD+1), IPHI, LAYER)
      CALL ECREI (TEMP)
      TNEW(INCRAD, IPHI, LAYER)=TEMP
C      ***** BOTTOM LAYER, MID-RADII *****
      DO 200 INCRAD=2, (NODES-1)
        RAD = RMIN + (INCRAD-1)*DR
C      ***** BOTTOM LAYER, MID-RADII, FIRST PHI *****
      IPHI = 1
        T1 = TOLD (INCRAD, IPHI, LAYER)
        T2 = TOLD (INCRAD, IPHI, (LAYER+1))
        T3 = TOLD (INCRAD, NPHI, LAYER)
        T4 = TOLD (INCRAD, (IPHI+1), LAYER)
        T5 = TOLD ((INCRAD-1), IPHI, LAYER)
        T6 = TOLD ((INCRAD+1), IPHI, LAYER)
        CALL ECREM (TEMP)
        TNEW(INCRAD, IPHI, LAYER) = TEMP
C      ***** BOTTOM LAYER, MID RADII, MID-PHIS *****
      DO 300 IPHI=2, (NPHI-1)
        T1 = TOLD (INCRAD, IPHI, LAYER)
        T2 = TOLD (INCRAD, IPHI, (LAYER+1))
        T3 = TOLD (INCRAD, (IPHI-1), LAYER)
        T4 = TOLD (INCRAD, (IPHI+1), LAYER)
        T5 = TOLD ((INCRAD-1), IPHI, LAYER)
        T6 = TOLD ((INCRAD+1), IPHI, LAYER)
        CALL ECREM (TEMP)
        TNEW(INCRAD, IPHI, LAYER) = TEMP
300      CONTINUE
C      ***** BOTTOM LAYER, MID RADII, LAST PHI *****
      IPHI = NPHI
        T1 = TOLD (INCRAD, IPHI, LAYER)
        T2 = TOLD (INCRAD, IPHI, (LAYER+1))
        T3 = TOLD (INCRAD, (IPHI-1), LAYER)
        T4 = TOLD (INCRAD, 1, LAYER)
        T5 = TOLD ((INCRAD-1), IPHI, LAYER)
        T6 = TOLD ((INCRAD+1), IPHI, LAYER)
        CALL ECREM (TEMP)
        TNEW(INCRAD, IPHI, LAYER) = TEMP
200      CONTINUE
C      ***** OUTER RADIUS *****
      INCRAD = NODES
      RAD = RMAX
C      ***** BOTTOM LAYER, OUTER RADIUS, FIRST PHI *****
      IPHI = 1
        T1 = TOLD (INCRAD, IPHI, LAYER)
        T2 = TOLD (INCRAD, IPHI, (LAYER+1))
        T3 = TOLD (INCRAD, NPHI, LAYER)
        T4 = TOLD (INCRAD, (IPHI+1), LAYER)
        T5 = TOLD ((INCRAD-1), IPHI, LAYER)
        CALL ECREO (TEMP)
        TNEW(INCRAD, IPHI, LAYER) = TEMP
C      ***** FIRST LAYER, OUTER RADIUS, MID-PHIS *****

```





```

DO 700 INCRAD=2, (NODES-1)
  RAD = RMIN + (INCRAD-1)*DR
C      ***** MID-LAYERS, MID-RADII, FIRST PHI *****
  IPHI = 1
    T1 = TOLD (INCRAD, IPHI, LAYER)
    T2 = TOLD (INCRAD, IPHI, (LAYER+1))
    T3 = TOLD (INCRAD, IPHI, (LAYER-1))
    T4 = TOLD (INCRAD, NPHI, LAYER)
    T5 = TOLD (INCRAD, (IPHI+1), LAYER)
    T6 = TOLD ((INCRAD-1), IPHI, LAYER)
    T7 = TOLD ((INCRAD+1), IPHI, LAYER)
    CALL ECRMM (TEMP)
    TNEW (INCRAD, IPHI, LAYER) = TEMP
C      ***** MID-LAYERS, MID-RADII, MID-PHIS *****
DO 800 IPHI=2, (NPHI-1)
  T1 = TOLD (INCRAD, IPHI, LAYER)
  T2 = TOLD (INCRAD, IPHI, (LAYER+1))
  T3 = TOLD (INCRAD, IPHI, (LAYER-1))
  T4 = TOLD (INCRAD, (IPHI-1), LAYER)
  T5 = TOLD (INCRAD, (IPHI+1), LAYER)
  T6 = TOLD ((INCRAD-1), IPHI, LAYER)
  T7 = TOLD ((INCRAD+1), IPHI, LAYER)
  CALL ECRMM (TEMP)
  TNEW (INCRAD, IPHI, LAYER) = TEMP
800  CONTINUE
C      ***** MID-LAYERS, MID-RADII, LAST PHI *****
  IPHI = NPHI
    T1 = TOLD (INCRAD, IPHI, LAYER)
    T2 = TOLD (INCRAD, IPHI, (LAYER+1))
    T3 = TOLD (INCRAD, IPHI, (LAYER-1))
    T4 = TOLD (INCRAD, (IPHI-1), LAYER)
    T5 = TOLD (INCRAD, 1, LAYER)
    T6 = TOLD ((INCRAD-1), IPHI, LAYER)
    T7 = TOLD ((INCRAD+1), IPHI, LAYER)
    CALL ECRMM (TEMP)
    TNEW (INCRAD, IPHI, LAYER) = TEMP
700  CONTINUE
C      **** MID-LAYERS, OUTER RADII ****
  INCRAD = NODES
  RAD = RMAX
C      ***** MID-LAYERS, OUTER RADIUS, FIRST PHI *****
  IPHI = 1
    T1 = TOLD (INCRAD, IPHI, LAYER)
    T2 = TOLD (INCRAD, IPHI, (LAYER+1))
    T3 = TOLD (INCRAD, IPHI, (LAYER-1))
    T4 = TOLD (INCRAD, NPHI, LAYER)
    T5 = TOLD (INCRAD, (IPHI+1), LAYER)
    T6 = TOLD ((INCRAD-1), IPHI, LAYER)
    CALL ECRMO (TEMP)
    TNEW (INCRAD, IPHI, LAYER) = TEMP
C      ***** MID-LAYERS, OUTER RADIUS, MID-PHIS *****
DO 900 IPHI=2, (NPHI-1)
  T1 = TOLD (INCRAD, IPHI, LAYER)
  T2 = TOLD (INCRAD, IPHI, (LAYER+1))
  T3 = TOLD (INCRAD, IPHI, (LAYER-1))

```



```

      T1 = TOLD(INCRAD, IPHI, LAYER)
      T2 = TOLD(INCRAD, IPHI, (LAYER-1))
      T3 = TOLD(INCRAD, NPHI, LAYER)
      T4 = TOLD(INCRAD, (IPHI+1), LAYER)
      T5 = TOLD((INCRAD-1), IPHI, LAYER)
      T6 = TOLD((INCRAD+1), IPHI, LAYER)
      CALL ECREM (TEMP)
      TNEW(INCRAD, IPHI, LAYER) = TEMP
C      ***** TOP LAYER, MID RADII, MID-PHIS *****
      DO 1200 IPHI=2, (NPHI-1)
        T1 = TOLD(INCRAD, IPHI, LAYER)
        T2 = TOLD(INCRAD, IPHI, (LAYER-1))
        T3 = TOLD(INCRAD, (IPHI-1), LAYER)
        T4 = TOLD(INCRAD, (IPHI+1), LAYER)
        T5 = TOLD((INCRAD-1), IPHI, LAYER)
        T6 = TOLD((INCRAD+1), IPHI, LAYER)
        CALL ECREM (TEMP)
        TNEW(INCRAD, IPHI, LAYER) = TEMP
1200      CONTINUE
C      ***** TOP LAYER, MID RADII, LAST PHI *****
      IPHI = NPHI
      T1 = TOLD(INCRAD, IPHI, LAYER)
      T2 = TOLD(INCRAD, IPHI, (LAYER-1))
      T3 = TOLD(INCRAD, (IPHI-1), LAYER)
      T4 = TOLD(INCRAD, 1, LAYER)
      T5 = TOLD((INCRAD-1), IPHI, LAYER)
      T6 = TOLD((INCRAD+1), IPHI, LAYER)
      CALL ECREM (TEMP)
      TNEW(INCRAD, IPHI, LAYER) = TEMP
1100      CONTINUE
C      **** TOP LAYER, OUTER RADIUS ****
      INCRAD = NODES
      RAD = RMAX
C      ***** TOP LAYER, OUTER RADIUS, FIRST PHI *****
      IPHI = 1
      T1 = TOLD(INCRAD, IPHI, LAYER)
      T2 = TOLD(INCRAD, IPHI, (LAYER-1))
      T3 = TOLD(INCRAD, NPHI, LAYER)
      T4 = TOLD(INCRAD, (IPHI+1), LAYER)
      T5 = TOLD((INCRAD-1), IPHI, LAYER)
      CALL ECREO (TEMP)
      TNEW(INCRAD, IPHI, LAYER) = TEMP
C      ***** TOP LAYER, OUTER RADIUS, MID-PHIS *****
      DO 1300 IPHI=2, (NPHI-1)
        T1 = TOLD(INCRAD, IPHI, LAYER)
        T2 = TOLD(INCRAD, IPHI, (LAYER-1))
        T3 = TOLD(INCRAD, (IPHI-1), LAYER)
        T4 = TOLD(INCRAD, (IPHI+1), LAYER)
        T5 = TOLD((INCRAD-1), IPHI, LAYER)
        CALL ECREO (TEMP)
        TNEW(INCRAD, IPHI, LAYER) = TEMP
1300      CONTINUE
C      ***** TOP LAYER, OUTER RADIUS, LAST PHI *****
      IPHI = NPHI
      T1 = TOLD(INCRAD, IPHI, LAYER)

```



```

C
C
C *****
C ECREM - THIS SUBROUTINE COMPUTES THE TEMPERATURES OF THE MID- *
C * RADIAL NODES FOR THE END LAYERS (NORTH, SOUTH) *
C * OF A RIGHT-CIRCULAR CYLINDER USING THE EXPLICIT METHOD *
C * OF FINITE DIFFERENCE ANALYSIS IN 3-D POLAR-CYLINDRICAL *
C * COORDINATES. *
C *****
C PROGRAM LINE 678
C SUBROUTINE ECREM (TNEW)
C IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C COMMON /CONUC/ DKR,DKP,DKZ
C COMMON /CONVEC/ HSUBN,HSUBS,HSUBI,HSUBO
C COMMON /GEOMET/ DPHI,DR,DZ,RAD,RMAX,RMIN,THICK,ZMAX,LPHI
C COMMON /HEAT/ ALFA,FO,QPSUBN,QPSUBS,QPSUBI,QPSUBO
C COMMON /MATSIZ/ NODES,NPHI,NLAYER
C COMMON /NPOSIT/ INCRAD,IPHI,LAYER
C COMMON /PARAMS/ CSUBP,RHO
C COMMON /RADN/ EPSUBN,EPSUBS,EPSUBI,EPSUBO,SIGMA
C COMMON /TEMPS/ T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,TINF,TSUR
C ***** BODY *****
C IF (LAYER.EQ.1) THEN
C   Q = QPSUBS
C   EPSE = EPSUBS
C   HE = HSUBS
C ELSEIF (LAYER.EQ.NLAYER) THEN
C   Q = QPSUBN
C   EPSE = EPSUBN
C   HE = HSUBN
C ENDIF
C D1 = (DKP*DZ**2)/(2*DKZ*RAD**2*DPHI**2)
C D2 = (DKR*(RAD-DR/2)*DZ**2)/(2*DKZ*RAD*DR**2)
C D3 = (DKR*(RAD+DR/2)*DZ**2)/(2*DKZ*RAD*DR**2)
C D4 = DZ/DKZ
C D5 = SIGMA*(T1+TSUR)*(T1**2+TSUR**2)
C D6 = 1 + 2*FO*D4*(HE+D5*EPSE)
C TNEW = 1/D6*(T1 + 2*FO*((T2-T1) + D1*(T3+T4-2*T1) + D2*(T5-T1)
C 1+ D3*(T6-T1) + D4*(Q + HE*TINF + D5*EPSE*TSUR))
998 RETURN
C END
C
C *****
C ECREO FORTRAN A - (REV. 11/14/88)
C *****
C * ECREO - THIS SUBROUTINE COMPUTES THE NODE TEMPERATURES AT *
C * THE OUTER SURFACE OF THE END LAYERS (NORTH, SOUTH) *
C * OF A RIGHT-CIRCULAR CYLINDER USING THE EXPLICIT METHOD *
C * OF FINITE DIFFERENCE ANALYSIS IN 3-D POLAR-CYLINDRICAL *
C * COORDINATES. *
C *****
C PROGRAM LINE 721
C SUBROUTINE ECREO (TNEW)

```











```

C *****
C * LNGFRM - SUBROUTINE TO PRINT FULL MATRIX OF TEMPERATURES *
C * FROM DATA FILES. *
C *****
C PROGRAM LINE 948
SUBROUTINE LNGFRM
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
COMMON /HEAT/ ALFA,FO,QPSUBN,QPSUBS,QPSUBI,QPSUBO
COMMON /TIMES/ DELTIM,DTMAX,FINTIM,LIMIT
DIMENSION TEMP(15,15,15)
OPEN (UNIT=11,FILE='ECRLNG')
OPEN (UNIT=14,FILE='ECRDAT')
READ (14,*) NODES,NPHI,NLAYER
WRITE (15,*) 'EXPLICIT - WITH CONDUCTION, CONVECTION AND RADIATION
1'
WRITE (15,*)
WRITE (15,*) 'DATA SAVED AS FILE:'
WRITE (15,*)
WRITE (15,*) 'CPU TIME:'
WRITE (15,*)
WRITE (11,*) 'GRID SIZE (RADIAL,PHI,Z DIRECTIONS):',NODES,NPHI,NLA
1YER
WRITE (11,*)
WRITE (11,*) 'THEORETICAL MAX TIMESTEP (EXPLICIT):',DTMAX WRITE
(11,*)
WRITE (11,*) 'SELECTED TIMESTEP (SEC):',DELTIM WRITE (11,*)
WRITE (11,*) 'GRID FOURIER NUMBER (Z,DELTIM DEPENDENT): ',FO WRITE
(11,*)
WRITE (11,*) 'SURFACE HEAT FLUX (W/M**2):',QPSUBO WRITE (11,*)
WRITE (11,*)
1 READ (14,*,END=998) TIME,DTIME
DO 100 K=1,NLAYER
DO 200 I=1,NODES
DO 300 J=1,NPHI
READ (14,*,END=997) TEMP(I,J,K)
300 CONTINUE
200 CONTINUE
100 CONTINUE
CALL PRMTAT (TIME,DTIME,TEMP,NODES,NPHI,NLAYER)
GOTO 1
997 WRITE(11,*) 'LNGFRM - EOF READ ERROR IN PROGRAM LINE 712'
WRITE(*,*) 'LNGFRM - EOF READ ERROR IN PROGRAM LINE 712'
998 REWIND 14
CLOSE (11)
RETURN
END
C
C *****
C PRMTAT FORTRAN A - (REV. 11/04/88)
C *****
C * PRMTAT - SUBROUTINE TO PRINT THREE DIMENDIONAL MATRIX *
C * TO FILE 11 (*LONG). *
C *****
C PROGRAM LINE 1000

```



```

COMMON /HEAT/ ALFA,FO,QPSUBN,QPSUBS,QPSUBI,QPSUBO
COMMON /TIMES/ DELTIM,DTMAX,FINTIM,LIMIT
DIMENSION TEMP(15,15,15)
OPEN(UNIT=13,FILE='ANALYT')
OPEN(UNIT=14,FILE='ECRDAT')
OPEN(UNIT=15,FILE='ECRERR')
IF (((QPSUBO.GT.1352).AND.(QPSUBO.LT.1354)).OR.((QPSUBO.GT.14999)
1.AND.(QPSUBO.LT.15001))) THEN
CONTINUE
ELSE
GOTO 998
ENDIF
C READ IN MATRIX SIZE
READ (14,*) NODES,NPHI,NLAYER
C WRITE HEADERS TO DATA FILE 15 (ERROR)
WRITE (15,*) 'EXPLICIT - WITH CONDUCTION, CONVECTION AND RADIATION
1'
WRITE (15,*)
WRITE (15,*) 'DATA SAVED AS FILE:'
WRITE (15,*)
WRITE (15,*) 'CPU TIME:'
WRITE (15,*)
WRITE (15,*) 'GRID SIZE (RADIAL,PHI,Z DIRECTIONS):',NODES,NPHI,NLAY
1ER
WRITE (15,*)
WRITE (15,*) 'SELECTED NODE FOR COMPARISON:',L,M,N WRITE (15,*)
WRITE (15,*) 'THEORETICAL MAX TIMESTEP (EXPLICIT):',DTMAX WRITE
(15,*)
WRITE (15,*) 'SELECTED TIMESTEP (SEC):',DELTIM WRITE (15,*)
WRITE (15,*) 'GRID FOURIER NUMBER (Z,DELTIM DEPENDENT): ',FO WRITE
(15,*)
WRITE (15,*) 'OUTER SURFACE HEAT FLUX (W/M**2):',QPSUBO WRITE
(15,*)
WRITE (15,*)
WRITE (15,15)
15 FORMAT (T33,'NODE',T48,'ANALYTIC',T64,'PERCENT') WRITE (15,25)
25 FORMAT (T5,'TIME',T17,'DELTIM',T30,'TEMPERATURE',T46,'TEMPERATURE'
1,T65,'ERROR'/)
C READ TIME, DELTIM AND TEMPERATURE MATRICES FROM FILE 14 (*DATA) 2
READ (14,*,END=998) TIME,DELTIM
DO 100 K=1,NLAYER
DO 200 I=1,NODES
DO 300 J=1,NPHI
READ (14,*,END=995) TEMP(I,J,K)
300 CONTINUE
200 CONTINUE
100 CONTINUE
C READ IN APPROPRIATE ANALYTIC SOLUTIONS FROM FILE 13 (ANALYT)
3 IF (QPSUBO.LT.1400.) THEN
READ (13,35,END=996) EXACT
35 FORMAT (T19,F14.8)
ELSE
READ (13,45,END=997) EXACT
45 FORMAT (T39,F14.8)
ENDIF

```

```

C      COMPUTE PERCENT ERROR AND WRITE TO FILE 15 (ERROR)
      PE = (EXACT-TEMP(L,M,N))/EXACT * 100.
      WRITE (15,55) TIME,DELTIM,TEMP(L,M,N),EXACT,PE
55      FORMAT (T1,F9.2,T15,F8.4,T26,F14.8,T41,F14.8,T61,F9.3)
      GOTO 2
995    WRITE (*,*) 'ERROR - EOF READ ERROR ON PROGRAM LINE 652'
      WRITE (15,*) 'ERROR - EOF READ ERROR ON PROGRAM LINE 652'
996    WRITE (*,*) 'ERROR - EOF READ ERROR ON PROGRAM LINE 658'
      WRITE (15,*) 'ERROR - EOF READ ERROR ON PROGRAM LINE 658'
997    WRITE (*,*) 'ERROR - EOF READ ERROR ON PROGRAM LINE 661'
      WRITE (15,*) 'ERROR - EOF READ ERROR ON PROGRAM LINE 661'
998    REWIND 14
      CLOSE (13)
      CLOSE (15)
      RETURN
      END

```

# APPENDIX F

## BRIAN'S METHOD CODE

```

C      BCR FORTRAN A - (REV. 12/08/88)
C      SATELLITE MODEL - VALIDATED 12/08/88
C      FOR FORTVS APPLICATION ONLY
C      TO EXECUTE TYPE 'LOAD BCR TIMER (START'
C      *****
C      * BCR - PROGRAM TO COMPUTE NODE TEMPERATURES FOR A RIGHT *
C      *      CIRCULAR CYLINDER SUBJECTED TO CONVECTIVE AND/OR *
C      *      RADIATIVE BOUNDARY CONDITIONS USING BRIAN'S      *
C      *      METHOD OF FINITE DIFFERENCE ANALYSIS.            *
C      *****
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C      DIMENSION TOLD(15,15,15),TNEW(15,15,15)
C      DIMENSION TSTAR(15,15,15),T2STAR(15,15,15)
C      DIMENSION T3STAR(15,15,15)
C      COMMON /ARRSIZ/ III,JJJ,KKK
C      COMMON /CONDOC/ DKR,DKP,DKZ
C      COMMON /CONVEC/ HSUBN,HSUBS,HSUBI,HSUBO
C      COMMON /GEOMET/ DPHI,DR,DZ,RAD,RMAX,RMIN,THICK,ZMAX,LPHI
C      COMMON /HEAT/   ALFA,FO,QPSUBN,QPSUBS,QPSUBI,QPSUBO
C      COMMON /MATSIZ/ NODES,NPHI,NLAYER
C      COMMON /PARAMS/ CSUBP,RHO
C      COMMON /RADN/   EPSUBN,EPSUBS,EPSUBI,EPSUBO,SIGMA
C      COMMON /TEMPS/  T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,TINF,TSUR
C      COMMON /TIMES/  DELTIM,DTMAX,FINTIM,LIMIT
C      START TIMER
C      CALL STIME
C      OPEN (UNIT=11,FILE='BCRLNG')
C      OPEN (UNIT=14,FILE='BCRDAT')
C      OPEN (UNIT=15,FILE='BCRCPU')
C      ***** USER DEFINED VARIABLES *****
C      DELTIM = 120.0D+00
C      FINTIM = 3000.D+00
C      IPRINT = 1
C      ACTUAL MATRIX DIMENSIONS
C      NODES = 10
C      NPHI = 4
C      NLAYER = 5
C      NODE SPECIFICATION FOR FILE ESHORT
C      MRAD = 10
C      MPHI = 2
C      MLAYER = 3
C      DECLARED ARRAY DIMENSIONS
C      III = 15
C      JJJ = 15
C      KKK = 15
C      ABSORPTIVITY COEFFICIENTS

```

```

      ALFAN = 0.4D+00
      ALFAS = 0.4D+00
      ALFAO = 0.4D+00
C     ALBEDO COEFFICIENT
      ALBEDO = 0.30D+00
C     THERMAL CONDUCTIVITIES
      DKR = 177.D+00
      DKP = 177.D+00
      DKZ = 177.D+00
C     EMISSIVITIES
      EPSUBN = 0.9D+00
      EPSUBS = 0.9D+00
      EPSUBI = 0.D+00
      EPSUBO = 0.9D+00
C     CONVECTION COEFFICIENTS
      HSUBN = 0.D+00
      HSUBS = 0.D+00
      HSUBI = 0.D+00
      HSUBO = 0.D+00
C     HEAT FLUXES
      SOLAR = 1353.D+00
      EIR   = 238.0D+00
      GEN = 1000.
      QPSUBI = 0.D+00
      QPSUBO = 0.D+00
      QPSUBN = 0.D+00
      QPSUBS = 0.D+00
C     GEOMETRY
      RMIN = 0.15244D+00
      RMAX = 0.22866D+00
      ZMAX = 1.D+00
C     VIEW FACTORS WITH THE EARTH
      VERTHN = 0.D+00
      VERTHS = 1.D+00
      VERTHI = 0.D+00
      VERTH0 = 0.D+00
C     VIEW FACTORS WITH SPACE
      VSPCN = 1.D+00
      VSPCS = 0.D+00
      VSPCI = 0.D+00
      VSPCO = 1.D+00
C     TEMPERATURES
      TABS = 273.D+00
      TINIT = 293.D+00
      TINF = 0.D+00
      TSUR = 4.D+00
C     MATERIAL PROPERTIES
      CSUBP = 875.0D+00
      RHO = 2270.0D+00
C     INITIALIZE OTHER VARIABLES
      ICOUNT = 0
      ITIME = 0
      ORBTIM = 0.D+00
      PI = 4.0D+00*DTAN(1.D+00)
      RAD = RMIN

```

```

SIGMA = 5.67D-08
TIME = 0.D+00
T1 = 0.D+00
C INITIALIZE MATRICES
CALL BMINIT (TOLD,TINIT)
CALL BMINIT (TSTAR,TINIT)
CALL BMINIT (T2STAR,TINIT)
CALL BMINIT (T3STAR,TINIT)
CALL BMINIT (TNEW,TINIT)
C SURFACE AREA CALCULATIONS
SAREAN = PI*(RMAX**2 - RMIN**2)
SAREAS = PI*(RMAX**2 - RMIN**2)
SAREAI = PI*(2*RMIN)*ZMAX
SAREAO = PI*(2*RMAX)*ZMAX
C INITIAL CALCULATIONS
ALFA = DKZ/(RHO*CSUBP)
DPHI = 2.*PI/NPHI
DR = (RMAX-RMIN)/(NODES-1)
DZ = ZMAX/(N_LAYER-1)
LPHI = 360/NPHI
QPSUBI = GEN/SAREAI
C ORBITAL PARAMETERS
PERIOD = 5440.D+00
ECLIPS = 2181.9D+00
OMEGA = 1.155D-03
TAU = 0.01*PERIOD/4
TIME1 = 0.D+00
TIME2 = PERIOD/4.D+00
TIME3 = 107.8D+00/360 * PERIOD
TIME4 = (270 - 17.8)/360 * PERIOD - 4*TAU
TIME5 = (270 - 17.8)/360 * PERIOD
TIME6 = 270/360 * PERIOD
C ESTABLISH EXPLICIT STABILITY REQUIREMENT (FOR REFERENCE ONLY)
CALL STABIL (DTMAX)
FO = ALFA*DELTIM/(DZ**2)
LIMIT = FINTIM/DELTIM + 500
C WRITE MATRIX DIMENSIONS TO FILE 14 (BCRDAT)
WRITE (14,*) NODES,NPHI,N_LAYER
C
C *** PRINT OPTION 3 - SAVES DESIGNATED NODE TEMPS AT SPECIFIED
C PRINT INTERVAL. USE FOR LONG PERIOD EVALS.
C
WRITE (11,*) 'BCRSAT'
WRITE (11,*)
WRITE (11,888)
888 FORMAT (T1,'TIME',T15,'DELTIM',T25,'LAYER1',T40,'LAYER3',T55,'LAYER'
1R')
WRITE (11,*)
C
CMMMMMMMMMMMMMMMMMMMM BEGIN TEMPERATURE COMPUTATIONS MMMMMMMMMMMMMMMMMMMM
C START CLOCK TIMER (MAINFRAME VERSIONS OF CODE ONLY)
WRITE (11,*) 'ENTERING LOOP TIME'
CALL GTIME (1,11)
DO 100 I=1,LIMIT
    ICOUNT=ICOUNT+1
    TIME=TIME+DELTIM

```



```

      ORBTIM = ORBTIM + DELTIM
      IF (ORBTIM.GE.PERIOD) ORBTIM = ORBTIM - PERIOD
C     SURFACE FLUX CALCULATIONS (TIME DEPENDENT)
      IF ((ORBTIM.GE.0).AND.(ORBTIM.LE.TIME2)) THEN
        QPSUBN = ALFAN*SOLAR*DCOS(OMEGA*ORBTIM)
        QPSUBS = ALFAS*(EIR + ALBEDO*SOLAR*DCOS(OMEGA*ORBTIM))
        QPSUBO = 0.5*ALFAO*SOLAR*DSIN(OMEGA*ORBTIM)
      ELSEIF ((ORBTIM.GT.TIME2).AND.(ORBTIM.LE.TIME3)) THEN
        QPSUBN = 0.
        QPSUBS = ALFAS*(EIR + (ALBEDO*DCOS(OMEGA*ORBTIM))*SOLAR)
        QPSUBO = 0.5*ALFAO*SOLAR*DSIN(OMEGA*ORBTIM)
      ELSEIF ((ORBTIM.GT.TIME3).AND.(ORBTIM.LE.TIME4)) THEN
        QPSUBN = 0.
        QPSUBS = ALFAS*EIR
        QPSUBO = 0.D+00
      ELSEIF ((ORBTIM.GT.TIME4).AND.(ORBTIM.LE.TIME5)) THEN
        QPSUBN = 0.D+00
        QPSUBS = ALFAS*(EIR
1              + DABS(ALBEDO*SOLAR*DSIN(OMEGA*ORBTIM))
1              *(1-DEXP(-(ORBTIM-TIME4)/TAU)))
        QPSUBO = 0.5*ALFAO*DABS(ALBEDO*SOLAR
1              + DSIN(OMEGA*ORBTIM)
1              *(1-DEXP(-(ORBTIM-TIME4))/TAU))
      ELSEIF ((ORBTIM.GT.TIME5).AND.(ORBTIM.LT.TIME6)) THEN
        QPSUBN = 0.D+00
        QPSUBS = ALFAS*(EIR
1              + DABS(ALBEDO*SOLAR*DSIN(OMEGA*ORBTIM)))
        QPSUBO = 0.5*ALFAO*DABS(SOLAR*DSIN(OMEGA*ORBTIM))
      ELSEIF (ORBTIM.GT.TIME6) THEN
        QPSUBN = ALFAN*DABS(SOLAR*DCOS(OMEGA*ORBTIM))
        QPSUBS = ALFAS*(EIR
1              + DABS(ALBEDO*SOLAR*DSIN(OMEGA*ORBTIM)))
        QPSUBO = 0.5*ALFAO*DABS(SOLAR*DSIN(OMEGA*ORBTIM))
      ENDIF
C     CALL MAIN SUBROUTINES
      CALL STEP1 (TOLD,TSTAR)
C     WRITE (11,*) 'TSTAR'
C     CALL PRMAT (TIME,DELTIM,TSTAR,NODES,NPHI,NLAYER)
      CALL STEP2 (TOLD,TSTAR,T2STAR)
C     WRITE (11,*) 'T2STAR'
C     CALL PRMAT (TIME,DELTIM,T2STAR,NODES,NPHI,NLAYER)
      CALL STEP3 (TOLD,TSTAR,T2STAR,T3STAR)
C     WRITE (11,*) 'T3STAR'
C     CALL PRMAT (TIME,DELTIM,T3STAR,NODES,NPHI,NLAYER)
      CALL STEP4 (TOLD,TSTAR,T2STAR,T3STAR,TNEW)
C     WRITE (11,*) 'TNEW'
C     CALL PRMAT (TIME,DELTIM,TNEW,NODES,NPHI,NLAYER)
C     MAP NEW TEMPERATURE MATRIX ONTO OLD TEMPERATURE MATRIX
      CALL MAP122 (TNEW,TOLD)
C
C     **** GENERATE PRINTOUT ****
      IF (ICOUNT.GE.IPRINT) THEN
        ICOUNT = 0
        WRITE (14,*) TIME,DELTIM
C     **** CONVERT TNEW TO EXCESS TEMPERATURE PRIOR TO PRINTING ****

```

```

      DO 200 LAYER=1,NLAYER
      DO 300 INCRAD=1,NODES
      DO 400 IPHI=1,NPHI
          TNEW(INCRAD,IPHI,LAYER) = TNEW(INCRAD,IPHI,LAYER 1) - TINIT
C     **** PRINT OPTION 1 - USE WITH CMPSHT,COMPAR,LNGFRM,SHTFRM,BCRERR
C     SHORT EVALUATION TIMES ONLY (<400 SEC)
C     WRITE (14,*) TNEW(INCRAD,IPHI,LAYER)
400      CONTINUE
300      CONTINUE
200      CONTINUE
C
C     *** PRINT OPTION 2 - SAVES COMPLETE MATRIX AT SPECIFIED PRINT
C     INTERVAL - SWITCH OFF ALL SUBROUTINES LISTED
C     UNDER PRINT OPTION 1 FIRST.
C     WRITE (11,*) 'BCR SATELLITE MODEL'
C     WRITE (*,*)
C     CALL PRMAT (TIME,DELTIM,TNEW,NODES,NPHI,NLAYER)
C
C     *** PRINT OPTION 3 - SAVES DESIGNATED NODE TEMPS AT SPECIFIED
C     PRINT INTERVAL. USE FOR LONG PERIOD EVALS.
C     WRITE (11,889) TIME,DELTIM,TNEW(5,1,1),TNEW(5,1,3),TNEW(5,1,5)
889    FORMAT (T1,F9.3,T15,F9.3,T25,F12.6,T40,F12.6,T55,F12.6)
C
C     ***** RE-SPECIFY PRINT INTERVALS *****
C     IF (TIME.GE.99.9) THEN
C         IPRINT = 200
C         DELTIM = 20.
C         FO = ALFA*DELTIM/(DZ**2)
C     ELSEIF (TIME.GE.9.9) THEN
C         IPRINT = 100
C         DELTIM = 10.
C         FO = ALFA*DELTIM/(DZ**2)
C     ELSEIF (TIME.GE.0.99) THEN
C         IPRINT = 10
C         DELTIM = 1.
C         FO = ALFA*DELTIM/(DZ**2)
C     ENDIF
C     ENDIF
C     TIME INTERVAL DELIMITER
C     IF (TIME.GE.FINTIM) GOTO 101
100    CONTINUE
C     STOP CLOCK TIMER (MAINFRAME VERSIONS OF CODE ONLY)
101    CONTINUE
CALL GTIME (1,11)
WRITE (11,*) 'EXITED LOOP'
CLOSE (14)
CLOSE (15)
C     WRITE APPROPRIATE DATA FILES
C     CALL CMPSHT
C     CALL SHTFRM (MRAD,MPHI,MLAYER)
C     CALL COMPAR
C     CALL ERROR (MRAD,MPHI,MLAYER)
C     CALL LNGFRM
CALL GTIME(1,11)
999    STOP

```







```

C
C      ***** INNER SURFACE, MID-PHI, TOP LAYER *****
      LAYER = NLayer
      T1 = TOLD (INCRAD, IPHI, LAYER)
      T2 = TOLD (INCRAD, (IPHI-1), LAYER)
      T3 = TOLD (INCRAD, (IPHI+1), LAYER)
      T4 = TOLD ((INCRAD+1), IPHI, LAYER)
      CALL BCREI1 (AA (LAYER), BB (LAYER), CC (LAYER), DD (LAYER))

C
C      ***** COMPUTE NODE TEMPERATURES *****
      CALL TRIDAG (1, NLayer, AA, BB, CC, DD, TEMP)
      DO 230 LAYER=1, NLayer
      TSTAR (INCRAD, IPHI, LAYER) = TEMP (LAYER)
230      CONTINUE
C
C      CONTINUE
C
C      ***** INNER SURFACE, LAST PHI *****
      IPHI = NPHI
C
C      ***** INNER SURFACE, LAST PHI, BOTTOM LAYER *****
      LAYER = 1
      T1 = TOLD (INCRAD, IPHI, LAYER)
      T2 = TOLD (INCRAD, (IPHI-1), LAYER)
      T3 = TOLD (INCRAD, 1, LAYER)
      T4 = TOLD ((INCRAD+1), IPHI, LAYER)
      CALL BCREI1 (AA (LAYER), BB (LAYER), CC (LAYER), DD (LAYER))

C
C      ***** INNER SURFACE, LAST PHI, MID-LAYERS *****
      DO 300 LAYER=2, (NLayer-1)
      T1 = TOLD (INCRAD, IPHI, LAYER)
      T2 = TOLD (INCRAD, (IPHI-1), LAYER)
      T3 = TOLD (INCRAD, 1, LAYER)
      T4 = TOLD ((INCRAD+1), IPHI, LAYER)
      CALL BCRM11 (AA (LAYER), BB (LAYER), CC (LAYER), DD (LAYER))
300      CONTINUE
C
C      ***** INNER SURFACE, LAST PHI, TOP LAYER *****
      LAYER = NLayer
      T1 = TOLD (INCRAD, IPHI, LAYER)
      T2 = TOLD (INCRAD, (IPHI-1), LAYER)
      T3 = TOLD (INCRAD, 1, LAYER)
      T4 = TOLD ((INCRAD+1), IPHI, LAYER)
      CALL BCREI1 (AA (LAYER), BB (LAYER), CC (LAYER), DD (LAYER))

C
C      ***** COMPUTE NODE TEMPERATURES *****
      CALL TRIDAG (1, NLayer, AA, BB, CC, DD, TEMP)
      DO 350 LAYER=1, NLayer
      TSTAR (INCRAD, IPHI, LAYER) = TEMP (LAYER)
350      CONTINUE
C
C
C      M***** MID-RADIUS NODES M*****
      DO 400 INCRAD=2, (NODES-1)
      RAD = RMIN + (INCRAD-1)*DR

```

```

C          ***** MID-RADII, FIRST PHI *****
IPHI = 1
C
C          ***** MID-RADII, FIRST PHI, BOTTOM LAYER *****
C          LAYER = 1
              T1 = TOLD(INCRAD, IPHI, LAYER)
              T2 = TOLD(INCRAD, NPHI, LAYER)
              T3 = TOLD(INCRAD, (IPHI+1), LAYER)
              T4 = TOLD((INCRAD-1), IPHI, LAYER)
              T5 = TOLD((INCRAD+1), IPHI, LAYER)
              CALL BCREM1 (AA(LAYER), BB(LAYER), CC(LAYER), DD(LAYER)) C
C          ***** MID-RADII, FIRST PHI, MID-LAYERS *****
DO 500 LAYER=2, (NLayer-1)
              T1 = TOLD(INCRAD, IPHI, LAYER)
              T2 = TOLD(INCRAD, NPHI, LAYER)
              T3 = TOLD(INCRAD, (IPHI+1), LAYER)
              T4 = TOLD((INCRAD-1), IPHI, LAYER)
              T5 = TOLD((INCRAD+1), IPHI, LAYER)
              CALL BCRMM1 (AA(LAYER), BB(LAYER), CC(LAYER), DD(LAYER))
500          CONTINUE
C
C          ***** MID-RADII, FIRST PHI, TOP LAYER *****
C          LAYER = NLayer
              T1 = TOLD(INCRAD, IPHI, LAYER)
              T2 = TOLD(INCRAD, NPHI, LAYER)
              T3 = TOLD(INCRAD, (IPHI+1), LAYER)
              T4 = TOLD((INCRAD-1), IPHI, LAYER)
              T5 = TOLD((INCRAD+1), IPHI, LAYER)
              CALL BCREM1 (AA(LAYER), BB(LAYER), CC(LAYER), DD(LAYER))
C
C          **** COMPUTE NODE TEMPERATURES ****
CALL TRIDAG (1, NLayer, AA, BB, CC, DD, TEMP)
DO 475 LAYER=1, NLayer
TSTAR(INCRAD, IPHI, LAYER) = TEMP(LAYER)
475          CONTINUE
C
C          ***** MID-RADII, MID-PHIS *****
DO 600, IPHI=2, (NPHI-1)
C
C          ***** MID-RADII, MID-PHIS, FIRST LAYER *****
C          LAYER = 1
              T1 = TOLD(INCRAD, IPHI, LAYER)
              T2 = TOLD(INCRAD, (IPHI-1), LAYER)
              T3 = TOLD(INCRAD, (IPHI+1), LAYER)
              T4 = TOLD((INCRAD-1), IPHI, LAYER)
              T5 = TOLD((INCRAD+1), IPHI, LAYER)
              CALL BCREM1 (AA(LAYER), BB(LAYER), CC(LAYER), DD(LAYER))
C
C          ***** MID-RADII, MID-PHIS, MID-LAYERS *****
DO 700 LAYER=2, (NLayer-1)
              T1 = TOLD(INCRAD, IPHI, LAYER)
              T2 = TOLD(INCRAD, (IPHI-1), LAYER)
              T3 = TOLD(INCRAD, (IPHI+1), LAYER)
              T4 = TOLD((INCRAD-1), IPHI, LAYER)
              T5 = TOLD((INCRAD+1), IPHI, LAYER)

```

```

              CALL BCRMM1 (AA(LAYER),BB(LAYER),CC(LAYER),DD(LAYER))
700      CONTINUE
C
C          ***** MID-RADII, MID-PHI, TOP LAYER *****
      LAYER = NLayer
          T1 = TOLD(INCRAD,IPHI,LAYER)
          T2 = TOLD(INCRAD,(IPHI-1),LAYER)
          T3 = TOLD(INCRAD,(IPHI+1),LAYER)
          T4 = TOLD((INCRAD-1),IPHI,LAYER)
          T5 = TOLD((INCRAD+1),IPHI,LAYER)
          CALL BCREM1 (AA(LAYER),BB(LAYER),CC(LAYER),DD(LAYER))
C          ***** COMPUTE NODE TEMPERATURES *****
      CALL TRIDAG (1,NLayer,AA,BB,CC,DD,TEMP)
      DO 575 LAYER=1,NLayer
          TSTAR(INCRAD,IPHI,LAYER) = TEMP(LAYER)
575      CONTINUE
600      CONTINUE
C
C          ***** MID-RADII, LAST PHI *****
      IPHI = NPHI
C
C          ***** MID-RADII, LAST PHI, FIRST LAYER *****
      LAYER = 1
          T1 = TOLD(INCRAD,IPHI,LAYER)
          T2 = TOLD(INCRAD,(IPHI-1),LAYER)
          T3 = TOLD(INCRAD,1,LAYER)
          T4 = TOLD((INCRAD-1),IPHI,LAYER)
          T5 = TOLD((INCRAD+1),IPHI,LAYER)
          CALL BCREM1 (AA(LAYER),BB(LAYER),CC(LAYER),DD(LAYER))
C
C          ***** MID-RADII, LAST PHI, MID-LAYERS *****
      DO 800 LAYER=2,(NLayer-1)
          T1 = TOLD(INCRAD,IPHI,LAYER)
          T2 = TOLD(INCRAD,(IPHI-1),LAYER)
          T3 = TOLD(INCRAD,1,LAYER)
          T4 = TOLD((INCRAD-1),IPHI,LAYER)
          T5 = TOLD((INCRAD+1),IPHI,LAYER)
          CALL BCRMM1 (AA(LAYER),BB(LAYER),CC(LAYER),DD(LAYER))
800      CONTINUE
C
C          ***** MID-RADII, LAST PHI, TOP LAYER *****
      LAYER = NLayer
          T1 = TOLD(INCRAD,IPHI,LAYER)
          T2 = TOLD(INCRAD,(IPHI-1),LAYER)
          T3 = TOLD(INCRAD,1,LAYER)
          T4 = TOLD((INCRAD-1),IPHI,LAYER)
          T5 = TOLD((INCRAD+1),IPHI,LAYER)
          CALL BCREM1 (AA(LAYER),BB(LAYER),CC(LAYER),DD(LAYER))
C
C          ***** COMPUTE NODE TEMPERATURES *****
      CALL TRIDAG (1,NLayer,AA,BB,CC,DD,TEMP)
      DO 675 LAYER=1,NLayer
          TSTAR(INCRAD,IPHI,LAYER) = TEMP(LAYER)
675      CONTINUE

```



```

400  CONTINUE
C
COOOOOOOOOOOOOOOOOOOOOOOO OUTER SURFACE NODES OOOOOOOOOOOOOOOOOOOOOOOO
INCRAD = NODES
RAD = RMAX
C
C      ***** OUTER RADIUS, FIRST PHI *****
      IPHI = 1
C
C      ***** OUTER RADIUS, FIRST PHI, BOTTOM LAYER *****
      LAYER = 1
          T1 = TOLD(INCRAD,IPHI,LAYER)
          T2 = TOLD(INCRAD,NPHI,LAYER)
          T3 = TOLD(INCRAD,(IPHI+1),LAYER)
          T4 = TOLD((INCRAD-1),IPHI,LAYER)
          CALL BCREO1 (AA(LAYER),BB(LAYER),CC(LAYER),DD(LAYER))
C
C      ***** OUTER RADIUS, FIRST PHI, MID-LAYERS *****
DO 900 LAYER=2,(NLayer-1)
          T1 = TOLD(INCRAD,IPHI,LAYER)
          T2 = TOLD(INCRAD,NPHI,LAYER)
          T3 = TOLD(INCRAD,(IPHI+1),LAYER)
          T4 = TOLD((INCRAD-1),IPHI,LAYER)
          CALL BCRMO1 (AA(LAYER),BB(LAYER),CC(LAYER),DD(LAYER))
900  CONTINUE
C
C      ***** OUTER RADIUS, FIRST PHI, TOP LAYER *****
      LAYER = NLayer
          T1 = TOLD(INCRAD,IPHI,LAYER)
          T2 = TOLD(INCRAD,NPHI,LAYER)
          T3 = TOLD(INCRAD,(IPHI+1),LAYER)
          T4 = TOLD((INCRAD-1),IPHI,LAYER)
          CALL BCREO1 (AA(LAYER),BB(LAYER),CC(LAYER),DD(LAYER))
C
C      **** COMPUTE NODE TEMPERATURES ***
      CALL TRIDAG (1,NLayer,AA,BB,CC,DD,TEMP)
      DO 775 LAYER=1,NLayer
          TSTAR(INCRAD,IPHI,LAYER) = TEMP(LAYER)
775  CONTINUE
C
C      ***** OUTER SURFACE, MID-PHIS *****
DO 1000 IPHI=2,(NPHI-1)
C
C      ***** OUTER RADIUS, MID-PHIS, FIRST LAYER *****
      LAYER = 1
          T1 = TOLD(INCRAD,IPHI,LAYER)
          T2 = TOLD(INCRAD,(IPHI-1),LAYER)
          T3 = TOLD(INCRAD,(IPHI+1),LAYER)
          T4 = TOLD((INCRAD-1),IPHI,LAYER)
          CALL BCREO1 (AA(LAYER),BB(LAYER),CC(LAYER),DD(LAYER))
C
C      ***** OUTER RADIUS, MID-PHIS, MID-LAYERS *****
DO 1100 LAYER=2,(NLayer-1)
          T1 = TOLD(INCRAD,IPHI,LAYER)
          T2 = TOLD(INCRAD,(IPHI-1),LAYER)

```

```

      T3 = TOLD(INCRAD, (IPHI+1), LAYER)
      T4 = TOLD((INCRAD-1), IPHI, LAYER)
      CALL BCRMO1 (AA(LAYER), BB(LAYER), CC(LAYER), DD(LAYER))
1100      CONTINUE
C
C      ***** OUTER RADIUS, MID-PHI, TOP LAYER *****
      LAYER = NLayer
      T1 = TOLD(INCRAD, IPHI, LAYER)
      T2 = TOLD(INCRAD, (IPHI-1), LAYER)
      T3 = TOLD(INCRAD, (IPHI+1), LAYER)
      T4 = TOLD((INCRAD-1), IPHI, LAYER)
      CALL BCREO1 (AA(LAYER), BB(LAYER), CC(LAYER), DD(LAYER))
C
C      **** COMPUTE NODE TEMPERATURES ****
      CALL TRIDAG (1, NLayer, AA, BB, CC, DD, TEMP)
      DO 875 LAYER=1, NLayer
      TSTAR(INCRAD, IPHI, LAYER) = TEMP(LAYER)
875      CONTINUE
1000      CONTINUE
C
C      ***** OUTER SURFACE, LAST PHI *****
      IPHI = NPHI
C
C      ***** OUTER RADIUS, LAST PHI, BOTTOM LAYER *****
      LAYER = 1
      T1 = TOLD(INCRAD, IPHI, LAYER)
      T2 = TOLD(INCRAD, (IPHI-1), LAYER)
      T3 = TOLD(INCRAD, 1, LAYER)
      T4 = TOLD((INCRAD-1), IPHI, LAYER)
      CALL BCREO1 (AA(LAYER), BB(LAYER), CC(LAYER), DD(LAYER))
C
C      ***** OUTER RADIUS, LAST PHI, MID-LAYERS *****
      DO 1200 LAYER=2, (NLayer-1)
      T1 = TOLD(INCRAD, IPHI, LAYER)
      T2 = TOLD(INCRAD, (IPHI-1), LAYER)
      T3 = TOLD(INCRAD, 1, LAYER)
      T4 = TOLD((INCRAD-1), IPHI, LAYER)
      CALL BCRMO1 (AA(LAYER), BB(LAYER), CC(LAYER), DD(LAYER))
1200      CONTINUE
C
C      ***** OUTER RADIUS, LAST PHI, TOP LAYER *****
      LAYER = NLayer
      T1 = TOLD(INCRAD, IPHI, LAYER)
      T2 = TOLD(INCRAD, (IPHI-1), LAYER)
      T3 = TOLD(INCRAD, 1, LAYER)
      T4 = TOLD((INCRAD-1), IPHI, LAYER)
      CALL BCREO1 (AA(LAYER), BB(LAYER), CC(LAYER), DD(LAYER))
C
C      **** COMPUTE NODE TEMPERATURES ****
      CALL TRIDAG (1, NLayer, AA, BB, CC, DD, TEMP)
      DO 975 LAYER=1, NLayer
      TSTAR(INCRAD, IPHI, LAYER) = TEMP(LAYER)
975      CONTINUE
998      RETURN
END

```



```

DO 300 IPHI=1,NPHI
    T1 = TOLD(INCRAD,IPHI,LAYER)
    T2 = TSTAR(INCRAD,IPHI,(LAYER+1))
    T3 = TSTAR(INCRAD,IPHI,LAYER)
    T4 = TOLD((INCRAD-1),IPHI,LAYER)
    T5 = TOLD((INCRAD+1),IPHI,LAYER)
    CALL BCREM2 (AA(IPHI),BB(IPHI),CC(IPHI),DD(IPHI))
300    CONTINUE
C
C      **** CONVERT TO TRIDIAGONAL FORM ****
BB(1) = BB(1)+AA(1)
AA(1) = 0.D+00
BB(NPHI) = BB(NPHI)+CC(NPHI)
CC(NPHI) = 0.D+00
C
C      **** COMPUTE NODE TEMPERATURES ****
CALL TRIDAG (1,NPHI,AA,BB,CC,DD,TEMP)
DO 375 IPHI=1,NPHI
    T2STAR(INCRAD,IPHI,LAYER) = TEMP(IPHI)
375    CONTINUE
200    CONTINUE
C
C      ***** FIRST LAYER, OUTER RADIUS *****
INCRAD = NODES
RAD = RMAX
DO 400 IPHI=1,NPHI
    T1 = TOLD(INCRAD,IPHI,LAYER)
    T2 = TSTAR(INCRAD,IPHI,(LAYER+1))
    T3 = TSTAR(INCRAD,IPHI,LAYER)
    T4 = TOLD((INCRAD-1),IPHI,LAYER)
    CALL BCREO2 (AA(IPHI),BB(IPHI),CC(IPHI),DD(IPHI))
400    CONTINUE
C
C      **** CONVERT TO TRIDIAGONAL FORM ****
BB(1) = BB(1)+AA(1)
AA(1) = 0.D+00
BB(NPHI) = BB(NPHI)+CC(NPHI)
CC(NPHI) = 0.D+00
C
C      **** COMPUTE NODE TEMPERATURES ****
CALL TRIDAG (1,NPHI,AA,BB,CC,DD,TEMP)
DO 475 IPHI=1,NPHI
    T2STAR(INCRAD,IPHI,LAYER) = TEMP(IPHI)
475    CONTINUE
C
C      CMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM MID-LAYERS MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
DO 500 LAYER=2,(NLAYER-1)
C
C      ***** MID LAYERS, INNER RADIUS *****
INCRAD = 1
RAD = RMIN
DO 600 IPHI=1,NPHI
    T1 = TOLD (INCRAD,IPHI,LAYER)
    T2 = TSTAR(INCRAD,IPHI,(LAYER+1))
    T3 = TSTAR(INCRAD,IPHI,(LAYER-1))

```

```

        T4 = TSTAR(INCRAD, IPHI, LAYER)
        T5 = TOLD((INCRAD+1), IPHI, LAYER)
        CALL BCRMI2 (AA(IPHI), BB(IPHI), CC(IPHI), DD(IPHI))
600    CONTINUE
C
C        **** CONVERT TO TRIDIAGONAL FORM ****
        BB(1) = BB(1)+AA(1)
        AA(1) = 0.D+00
        BB(NPHI) = BB(NPHI)+CC(NPHI)
        CC(NPHI) = 0.D+00
C
C        **** COMPUTE NODE TEMPERATURES ****
        CALL TRIDAG (1, NPHI, AA, BB, CC, DD, TEMP)
        DO 675 IPHI=1, NPHI
            T2STAR(INCRAD, IPHI, LAYER) = TEMP(IPHI)
675    CONTINUE
C
C        ***** MID-LAYERS, MID-RADII *****
        DO 700 INCRAD=2, (NODES-1)
            RAD=RMIN+(INCRAD-1)*DR
            DO 800 IPHI=1, NPHI
                T1 = TOLD(INCRAD, IPHI, LAYER)
                T2 = TSTAR(INCRAD, IPHI, (LAYER+1))
                T3 = TSTAR(INCRAD, IPHI, (LAYER-1))
                T4 = TSTAR(INCRAD, IPHI, LAYER)
                T5 = TOLD((INCRAD-1), IPHI, LAYER)
                T6 = TOLD((INCRAD+1), IPHI, LAYER)
                CALL BCRMM2 (AA(IPHI), BB(IPHI), CC(IPHI), DD(IPHI))
800    CONTINUE
C
C        **** CONVERT TO TRIDIAGONAL FORM ****
        BB(1) = BB(1)+AA(1)
        AA(1) = 0.D+00
        BB(NPHI) = BB(NPHI)+CC(NPHI)
        CC(NPHI) = 0.D+00
C
C        **** COMPUTE NODE TEMPERATURES ****
        CALL TRIDAG (1, NPHI, AA, BB, CC, DD, TEMP)
        DO 875 IPHI=1, NPHI
            T2STAR(INCRAD, IPHI, LAYER) = TEMP(IPHI)
875    CONTINUE
700    CONTINUE
C
C        ***** MID-LAYERS, OUTER RADIUS *****
        INCRAD = NODES
        RAD = RMAX
        DO 900 IPHI=1, NPHI
            T1 = TOLD(INCRAD, IPHI, LAYER)
            T2 = TSTAR(INCRAD, IPHI, (LAYER+1))
            T3 = TSTAR(INCRAD, IPHI, (LAYER-1))
            T4 = TSTAR(INCRAD, IPHI, LAYER)
            T5 = TOLD((INCRAD-1), IPHI, LAYER)
            CALL BCRMO2 (AA(IPHI), BB(IPHI), CC(IPHI), DD(IPHI))
900    CONTINUE
C

```

[illegible]







```

C
C      ***** BOTTOM LAYER, MID-RAYS, INNER NODE *****
INCRAD = 1
RAD    = RMIN
      T1 = TOLD (INCRAD, IPHI, LAYER)
      T2 = TSTAR (INCRAD, IPHI, (LAYER+1))
      T3 = TSTAR (INCRAD, IPHI, LAYER)
      T4 = T2STAR (INCRAD, (IPHI-1), LAYER)
      T5 = T2STAR (INCRAD, (IPHI+1), LAYER)
      T6 = T2STAR (INCRAD, IPHI, LAYER)
      CALL BCREI3 (AA (INCRAD), BB (INCRAD), CC (INCRAD), DD (INCRAD))

C
C      ***** BOTTOM LAYER, MID-RAYS, MID-NODES *****
DO 300 INCRAD=2, (NODES-1)
RAD = RMIN + (INCRAD-1)*DR
      T1 = TOLD (INCRAD, IPHI, LAYER)
      T2 = TSTAR (INCRAD, IPHI, (LAYER+1))
      T3 = TSTAR (INCRAD, IPHI, LAYER)
      T4 = T2STAR (INCRAD, (IPHI-1), LAYER)
      T5 = T2STAR (INCRAD, (IPHI+1), LAYER)
      T6 = T2STAR (INCRAD, IPHI, LAYER)
      CALL BCREM3 (AA (INCRAD), BB (INCRAD), CC (INCRAD), DD (INCRAD))
300    CONTINUE

C
C      ***** BOTTOM LAYER, MID-RAYS, OUTER NODE *****
RAD = RMAX
INCRAD = NODES
      T1 = TOLD (INCRAD, IPHI, LAYER)
      T2 = TSTAR (INCRAD, IPHI, (LAYER+1))
      T3 = TSTAR (INCRAD, IPHI, LAYER)
      T4 = T2STAR (INCRAD, (IPHI-1), LAYER)
      T5 = T2STAR (INCRAD, (IPHI+1), LAYER)
      T6 = T2STAR (INCRAD, IPHI, LAYER)
      CALL BCREO3 (AA (INCRAD), BB (INCRAD), CC (INCRAD), DD (INCRAD))

C
C      ***** COMPUTE NODE TEMPERATURES *****
CALL TRIDAG (1, NODES, AA, BB, CC, DD, TEMP)
DO 375 INCRAD=1, NODES
T3STAR (INCRAD, IPHI, LAYER) = TEMP (INCRAD)
375    CONTINUE
200  CONTINUE

C
C      ***** BOTTOM LAYER, LAST RAY *****
IPHI = NPHI

C
C      ***** BOTTOM LAYER, LAST RAY, INNER NODE *****
INCRAD = 1
RAD    = RMIN
      T1 = TOLD (INCRAD, IPHI, LAYER)
      T2 = TSTAR (INCRAD, IPHI, (LAYER+1))
      T3 = TSTAR (INCRAD, IPHI, LAYER)
      T4 = T2STAR (INCRAD, (IPHI-1), LAYER)
      T5 = T2STAR (INCRAD, 1, LAYER)
      T6 = T2STAR (INCRAD, IPHI, LAYER)
      CALL BCREI3 (AA (INCRAD), BB (INCRAD), CC (INCRAD), DD (INCRAD))

```



```

        T5 = T2STAR(INCRAD,NPHI,LAYER)
        T6 = T2STAR(INCRAD,(IPHI+1),LAYER)
        T7 = T2STAR(INCRAD,IPHI,LAYER)
        CALL BCRMM3 (AA(INCRAD),BB(INCRAD),CC(INCRAD),DD(INCRAD))
600      CONTINUE
C
C      ***** INTERMEDIATE LAYERS, FIRST RAY, OUTER NODE *****
      INCRAD = NODES
      RAD = RMAX
        T1 = TOLD(INCRAD,IPHI,LAYER)
        T2 = TSTAR(INCRAD,IPHI,(LAYER+1))
        T3 = TSTAR(INCRAD,IPHI,(LAYER-1))
        T4 = TSTAR(INCRAD,IPHI,LAYER)
        T5 = T2STAR(INCRAD,NPHI,LAYER)
        T6 = T2STAR(INCRAD,(IPHI+1),LAYER)
        T7 = T2STAR(INCRAD,IPHI,LAYER)
        CALL BCRMO3 (AA(INCRAD),BB(INCRAD),CC(INCRAD),DD(INCRAD))
C
C      **** COMPUTE NODE TEMPERATURES ****
      CALL TRIDAG (1,NODES,AA,BB,CC,DD,TEMP)
      DO 675 INCRAD=1,NODES
        T3STAR(INCRAD,IPHI,LAYER) = TEMP(INCRAD)
675      CONTINUE
C
C      ***** INTERMEDIATE LAYERS, MID-RAYS *****
      DO 700 IPHI=2,(NPHI-1)
C
C      ***** INTERMEDIATE LAYERS, MID-RAYS, INNER NODE *****
      INCRAD = 1
      RAD = RMIN
        T1 = TOLD(INCRAD,IPHI,LAYER)
        T2 = TSTAR(INCRAD,IPHI,(LAYER+1))
        T3 = TSTAR(INCRAD,IPHI,(LAYER-1))
        T4 = TSTAR(INCRAD,IPHI,LAYER)
        T5 = T2STAR(INCRAD,(IPHI-1),LAYER)
        T6 = T2STAR(INCRAD,(IPHI+1),LAYER)
        T7 = T2STAR(INCRAD,IPHI,LAYER)
        CALL BCRMI3 (AA(INCRAD),BB(INCRAD),CC(INCRAD),DD(INCRAD))
C
C      ***** INTERMEDIATE LAYERS, MID-RAYS, MID-NODES *****
      DO 800 INCRAD=2,(NODES-1)
        RAD = RMIN + (INCRAD-1)*DR
        T1 = TOLD(INCRAD,IPHI,LAYER)
        T2 = TSTAR(INCRAD,IPHI,(LAYER+1))
        T3 = TSTAR(INCRAD,IPHI,(LAYER-1))
        T4 = TSTAR(INCRAD,IPHI,LAYER)
        T5 = T2STAR(INCRAD,(IPHI-1),LAYER)
        T6 = T2STAR(INCRAD,(IPHI+1),LAYER)
        T7 = T2STAR(INCRAD,IPHI,LAYER)
        CALL BCRMM3 (AA(INCRAD),BB(INCRAD),CC(INCRAD),DD(INCRAD))
800      CONTINUE
C
C      ***** INTERMEDIATE LAYERS, MID-RAYS, OUTER NODE *****
      INCRAD = NODES
      RAD = RMAX

```

```

      T1 = TOLD(INCRAD, IPHI, LAYER)
      T2 = TSTAR(INCRAD, IPHI, (LAYER+1))
      T3 = TSTAR(INCRAD, IPHI, (LAYER-1))
      T4 = TSTAR(INCRAD, IPHI, LAYER)
      T5 = T2STAR(INCRAD, (IPHI-1), LAYER)
      T6 = T2STAR(INCRAD, (IPHI+1), LAYER)
      T7 = T2STAR(INCRAD, IPHI, LAYER)
      CALL BCRMO3 (AA(INCRAD), BB(INCRAD), CC(INCRAD), DD(INCRAD))

C
C      **** COMPUTE NODE TEMPERATURES ****
      CALL TRIDAG (1, NODES, AA, BB, CC, DD, TEMP)
      DO 875 INCRAD=1, NODES
      T3STAR(INCRAD, IPHI, LAYER) = TEMP(INCRAD)
875      CONTINUE
700      CONTINUE
C
C      ***** INTERMEDIATE LAYERS, LAST RAY *****
      IPHI = NPHI
C
C      ***** INTERMEDIATE LAYERS, LAST RAY, INNER NODE *****
      INCRAD = 1
      RAD = RMIN
      T1 = TOLD(INCRAD, IPHI, LAYER)
      T2 = TSTAR(INCRAD, IPHI, (LAYER+1))
      T3 = TSTAR(INCRAD, IPHI, (LAYER-1))
      T4 = TSTAR(INCRAD, IPHI, LAYER)
      T5 = T2STAR(INCRAD, (IPHI-1), LAYER)
      T6 = T2STAR(INCRAD, 1, LAYER)
      T7 = T2STAR(INCRAD, IPHI, LAYER)
      CALL BCRMI3 (AA(INCRAD), BB(INCRAD), CC(INCRAD), DD(INCRAD))
C
C      ***** INTERMEDIATE LAYERS, LAST RAY, MID-NODES *****
      DO 900 INCRAD=2, (NODES-1)
      RAD = RMIN + (INCRAD-1)*DR
      T1 = TOLD(INCRAD, IPHI, LAYER)
      T2 = TSTAR(INCRAD, IPHI, (LAYER+1))
      T3 = TSTAR(INCRAD, IPHI, (LAYER-1))
      T4 = TSTAR(INCRAD, IPHI, LAYER)
      T5 = T2STAR(INCRAD, (IPHI-1), LAYER)
      T6 = T2STAR(INCRAD, 1, LAYER)
      T7 = T2STAR(INCRAD, IPHI, LAYER)
      CALL BCRMM3 (AA(INCRAD), BB(INCRAD), CC(INCRAD), DD(INCRAD))
900      CONTINUE
C
C      ***** INTERMEDIATE LAYERS, LAST RAY, OUTER NODE *****
      INCRAD = NODES
      RAD = RMAX
      T1 = TOLD(INCRAD, IPHI, LAYER)
      T2 = TSTAR(INCRAD, IPHI, (LAYER+1))
      T3 = TSTAR(INCRAD, IPHI, (LAYER-1))
      T4 = TSTAR(INCRAD, IPHI, LAYER)
      T5 = T2STAR(INCRAD, (IPHI-1), LAYER)
      T6 = T2STAR(INCRAD, 1, LAYER)
      T7 = T2STAR(INCRAD, IPHI, LAYER)
      CALL BCRMO3 (AA(INCRAD), BB(INCRAD), CC(INCRAD), DD(INCRAD))

```

```

C
C      **** COMPUTE NODE TEMPERATURES ****
CALL TRIDAG (1,NODES,AA,BB,CC,DD,TEMP)
DO 975 INCRAD=1,NODES
  T3STAR(INCRAD,IPHI,LAYER) = TEMP(INCRAD)
975      CONTINUE
500      CONTINUE
C
CTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT TOP LAYER TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
LAYER = NLAYER
C
C      ***** TOP LAYER, FIRST RAY *****
IPHI = 1
C
C      ***** TOP LAYER, FIRST RAY, INNER NODE *****
INCRAD = 1
RAD = RMIN
  T1 = TOLD(INCRAD,IPHI,LAYER)
  T2 = TSTAR(INCRAD,IPHI,(LAYER-1))
  T3 = TSTAR(INCRAD,IPHI,LAYER)
  T4 = T2STAR(INCRAD,NPHI,LAYER)
  T5 = T2STAR(INCRAD,(IPHI+1),LAYER)
  T6 = T2STAR(INCRAD,IPHI,LAYER)
CALL BCREI3 (AA(INCRAD),BB(INCRAD),CC(INCRAD),DD(INCRAD))
C
C      ***** TOP LAYER, FIRST RAY, MID-NODES *****
DO 1000 INCRAD=2,(NODES-1)
  RAD = RMIN + (INCRAD-1)*DR
    T1 = TOLD(INCRAD,IPHI,LAYER)
    T2 = TSTAR(INCRAD,IPHI,(LAYER-1))
    T3 = TSTAR(INCRAD,IPHI,LAYER)
    T4 = T2STAR(INCRAD,NPHI,LAYER)
    T5 = T2STAR(INCRAD,(IPHI+1),LAYER)
    T6 = T2STAR(INCRAD,IPHI,LAYER)
    CALL BCREM3 (AA(INCRAD),BB(INCRAD),CC(INCRAD),DD(INCRAD))
1000      CONTINUE
C
C      ***** TOP LAYER, FIRST RAY, OUTER NODE *****
INCRAD = NODES
RAD = RMAX
  T1 = TOLD(INCRAD,IPHI,LAYER)
  T2 = TSTAR(INCRAD,IPHI,(LAYER-1))
  T3 = TSTAR(INCRAD,IPHI,LAYER)
  T4 = T2STAR(INCRAD,NPHI,LAYER)
  T5 = T2STAR(INCRAD,(IPHI+1),LAYER)
  T6 = T2STAR(INCRAD,IPHI,LAYER)
CALL BCREO3 (AA(INCRAD),BB(INCRAD),CC(INCRAD),DD(INCRAD))
C
C      **** COMPUTE NODE TEMPERATURES ****
CALL TRIDAG (1,NODES,AA,BB,CC,DD,TEMP)
DO 1075 INCRAD=1,NODES
  T3STAR(INCRAD,IPHI,LAYER) = TEMP(INCRAD)
1075      CONTINUE
C
C      ***** TOP LAYER, MID-RAYS *****

```

```

DO 1100 IPHI=2, (NPHI-1)
C
C      ***** TOP LAYER, MID-RAYS, INNER NODE *****
INCRAD = 1
RAD    = RMIN
      T1 = TOLD (INCRAD, IPHI, LAYER)
      T2 = TSTAR (INCRAD, IPHI, (LAYER-1))
      T3 = TSTAR (INCRAD, IPHI, LAYER)
      T4 = T2STAR (INCRAD, (IPHI-1), LAYER)
      T5 = T2STAR (INCRAD, (IPHI+1), LAYER)
      T6 = T2STAR (INCRAD, IPHI, LAYER)
      CALL BCREI3 (AA (INCRAD), BB (INCRAD), CC (INCRAD), DD (INCRAD))
C
C      ***** TOP LAYER, MID-RAYS, MID-NODES *****
DO 1200 INCRAD=2, (NODES-1)
RAD = RMIN + (INCRAD-1)*DR
      T1 = TOLD (INCRAD, IPHI, LAYER)
      T2 = TSTAR (INCRAD, IPHI, (LAYER-1))
      T3 = TSTAR (INCRAD, IPHI, LAYER)
      T4 = T2STAR (INCRAD, (IPHI-1), LAYER)
      T5 = T2STAR (INCRAD, (IPHI+1), LAYER)
      T6 = T2STAR (INCRAD, IPHI, LAYER)
      CALL BCREM3 (AA (INCRAD), BB (INCRAD), CC (INCRAD), DD (INCRAD))
1200 CONTINUE
C
C      ***** TOP LAYER, MID-RAYS, OUTER NODE *****
INCRAD = NODES
RAD = RMAX
      T1 = TOLD (INCRAD, IPHI, LAYER)
      T2 = TSTAR (INCRAD, IPHI, (LAYER-1))
      T3 = TSTAR (INCRAD, IPHI, LAYER)
      T4 = T2STAR (INCRAD, (IPHI-1), LAYER)
      T5 = T2STAR (INCRAD, (IPHI+1), LAYER)
      T6 = T2STAR (INCRAD, IPHI, LAYER)
      CALL BCREO3 (AA (INCRAD), BB (INCRAD), CC (INCRAD), DD (INCRAD))
C
C      ***** COMPUTE NODE TEMPERATURES *****
CALL TRIDAG (1, NODES, AA, BB, CC, DD, TEMP)
DO 1275 INCRAD=1, NODES
T3STAR (INCRAD, IPHI, LAYER) = TEMP (INCRAD)
1275 CONTINUE
1100 CONTINUE
C
C      ***** TOP LAYER, LAST RAY *****
IPHI = NPHI
C
C      ***** TOP LAYER, LAST RAY, INNER NODE *****
INCRAD = 1
RAD = RMIN
      T1 = TOLD (INCRAD, IPHI, LAYER)
      T2 = TSTAR (INCRAD, IPHI, (LAYER-1))
      T3 = TSTAR (INCRAD, IPHI, LAYER)
      T4 = T2STAR (INCRAD, (IPHI-1), LAYER)
      T5 = T2STAR (INCRAD, 1, LAYER)
      T6 = T2STAR (INCRAD, IPHI, LAYER)

```



```

COMMON /RADN/ EPSUBN,EPSUBS,EPSUBI,EPSUBO,SIGMA
COMMON /TEMPS/ T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,TINF,TSUR
DIMENSION TOLD(15,15,15),TSTAR(15,15,15),T2STAR(15,15,15)
DIMENSION T3STAR(15,15,15),TNEW(15,15,15)

C
CBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
LAYER = 1

C
C          **** BOTTOM LAYER, INNER RADIUS ****

INCRAD = 1
RAD = RMIN

C
C ***** BOTTOM LAYER, INNER RADIUS, FIRST PHI *****
IPHI = 1
      T1 = TOLD(INCRAD,IPHI,LAYER)
      T2 = TSTAR(INCRAD,IPHI,(LAYER+1))
      T3 = TSTAR(INCRAD,IPHI,LAYER)
      T4 = T2STAR(INCRAD,NPHI,LAYER)
      T5 = T2STAR(INCRAD,(IPHI+1),LAYER)
      T6 = T2STAR(INCRAD,IPHI,LAYER)
      T7 = T3STAR((INCRAD+1),IPHI,LAYER)
      T8 = T3STAR(INCRAD,IPHI,LAYER)
      CALL BCREI4 (TEMP)
      TNEW(INCRAD,IPHI,LAYER) = TEMP

C
C          ***** BOTTOM LAYER, INNER RADIUS, MID-PHIS *****
DO 100 IPHI=2,(NPHI-1)
      T1 = TOLD(INCRAD,IPHI,LAYER)
      T2 = TSTAR(INCRAD,IPHI,(LAYER+1))
      T3 = TSTAR(INCRAD,IPHI,LAYER)
      T4 = T2STAR(INCRAD,(IPHI-1),LAYER)
      T5 = T2STAR(INCRAD,(IPHI+1),LAYER)
      T6 = T2STAR(INCRAD,IPHI,LAYER)
      T7 = T3STAR((INCRAD+1),IPHI,LAYER)
      T8 = T3STAR(INCRAD,IPHI,LAYER)
      CALL BCREI4 (TEMP)
      TNEW(INCRAD,IPHI,LAYER) = TEMP

100 CONTINUE

C
C          ***** BOTTOM LAYER, INNER RADIUS, LAST PHI *****
IPHI = NPHI
      T1 = TOLD(INCRAD,IPHI,LAYER)
      T2 = TSTAR(INCRAD,IPHI,(LAYER+1))
      T3 = TSTAR(INCRAD,IPHI,LAYER)
      T4 = T2STAR(INCRAD,(IPHI-1),LAYER)
      T5 = T2STAR(INCRAD,1,LAYER)
      T6 = T2STAR(INCRAD,IPHI,LAYER)
      T7 = T3STAR((INCRAD+1),IPHI,LAYER)
      T8 = T3STAR(INCRAD,IPHI,LAYER)
      CALL BCREI4 (TEMP)
      TNEW(INCRAD,IPHI,LAYER) = TEMP

C
C          ***** BOTTOM LAYER, MID-RADII *****
DO 200 INCRAD=2,(NODES-1)
RAD = RMIN + (INCRAD-1)*DR

```



```

C
C      ***** BOTTOM LAYER, MID-RADII, FIRST PHI *****
C      IPHI = 1
C          T1 = TOLD(INCRAD, IPHI, LAYER)
C          T2 = TSTAR(INCRAD, IPHI, (LAYER+1))
C          T3 = TSTAR(INCRAD, IPHI, LAYER)
C          T4 = T2STAR(INCRAD, NPHI, LAYER)
C          T5 = T2STAR(INCRAD, (IPHI+1), LAYER)
C          T6 = T2STAR(INCRAD, IPHI, LAYER)
C          T7 = T3STAR((INCRAD-1), IPHI, LAYER)
C          T8 = T3STAR((INCRAD+1), IPHI, LAYER)
C          T9 = T3STAR(INCRAD, IPHI, LAYER)
C          CALL BCREM4 (TEMP)
C          TNEW(INCRAD, IPHI, LAYER) = TEMP
C
C      ***** BOTTOM LAYER, MID RADII, MID-PHIS *****
C      DO 300 IPHI=2, (NPHI-1)
C          T1 = TOLD(INCRAD, IPHI, LAYER)
C          T2 = TSTAR(INCRAD, IPHI, (LAYER+1))
C          T3 = TSTAR(INCRAD, IPHI, LAYER)
C          T4 = T2STAR(INCRAD, (IPHI-1), LAYER)
C          T5 = T2STAR(INCRAD, (IPHI+1), LAYER)
C          T6 = T2STAR(INCRAD, IPHI, LAYER)
C          T7 = T3STAR((INCRAD-1), IPHI, LAYER)
C          T8 = T3STAR((INCRAD+1), IPHI, LAYER)
C          T9 = T3STAR(INCRAD, IPHI, LAYER)
C          CALL BCREM4 (TEMP)
C          TNEW(INCRAD, IPHI, LAYER) = TEMP
300      CONTINUE
C
C      ***** BOTTOM LAYER, MID RADII, LAST PHI *****
C      IPHI = NPHI
C          T1 = TOLD(INCRAD, IPHI, LAYER)
C          T2 = TSTAR(INCRAD, IPHI, (LAYER+1))
C          T3 = TSTAR(INCRAD, IPHI, LAYER)
C          T4 = T2STAR(INCRAD, (IPHI-1), LAYER)
C          T5 = T2STAR(INCRAD, 1, LAYER)
C          T6 = T2STAR(INCRAD, IPHI, LAYER)
C          T7 = T3STAR((INCRAD-1), IPHI, LAYER)
C          T8 = T3STAR((INCRAD+1), IPHI, LAYER)
C          T9 = T3STAR(INCRAD, IPHI, LAYER)
C          CALL BCREM4 (TEMP)
C          TNEW(INCRAD, IPHI, LAYER) = TEMP
200      CONTINUE
C
C          ***** OUTER RADIUS *****
C          INCRAD = NODES
C          RAD = RMAX
C
C      ***** BOTTOM LAYER, OUTER RADIUS, FIRST PHI *****
C      IPHI = 1
C          T1 = TOLD(INCRAD, IPHI, LAYER)
C          T2 = TSTAR(INCRAD, IPHI, (LAYER+1))
C          T3 = TSTAR(INCRAD, IPHI, LAYER)
C          T4 = T2STAR(INCRAD, NPHI, LAYER)

```

```

      T5 = T2STAR(INCRAD, (IPHI+1), LAYER)
      T6 = T2STAR(INCRAD, IPHI, LAYER)
      T7 = T3STAR((INCRAD-1), IPHI, LAYER)
      T8 = T3STAR(INCRAD, IPHI, LAYER)
      CALL BCREO4 (TEMP)
      TNEW(INCRAD, IPHI, LAYER) = TEMP
C
C      ***** FIRST LAYER, OUTER RADIUS, MID-PHI *****
DO 400 IPHI=2, (NPHI-1)
      T1 = TOLD(INCRAD, IPHI, LAYER)
      T2 = TSTAR(INCRAD, IPHI, (LAYER+1))
      T3 = TSTAR(INCRAD, IPHI, LAYER)
      T4 = T2STAR(INCRAD, (IPHI-1), LAYER)
      T5 = T2STAR(INCRAD, (IPHI+1), LAYER)
      T6 = T2STAR(INCRAD, IPHI, LAYER)
      T7 = T3STAR((INCRAD-1), IPHI, LAYER)
      T8 = T3STAR(INCRAD, IPHI, LAYER)

CALL BCREO4 (TEMP)
TNEW(INCRAD, IPHI, LAYER) = TEMP
400      CONTINUE
C
C      ***** BOTTOM LAYER, OUTER RADIUS, LAST PHI *****
      IPHI = NPHI
      T1 = TOLD(INCRAD, IPHI, LAYER)
      T2 = TSTAR(INCRAD, IPHI, (LAYER+1))
      T3 = TSTAR(INCRAD, IPHI, LAYER)
      T4 = T2STAR(INCRAD, (IPHI-1), LAYER)
      T5 = T2STAR(INCRAD, 1, LAYER)
      T6 = T2STAR(INCRAD, IPHI, LAYER)
      T7 = T3STAR((INCRAD-1), IPHI, LAYER)
      T8 = T3STAR(INCRAD, IPHI, LAYER)
      CALL BCREO4 (TEMP)
      TNEW(INCRAD, IPHI, LAYER) = TEMP
C
C      CMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM MID-LAYERS MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
DO 500 LAYER=2, (NLAYER-1)
C
C      ***** MID-LAYERS, INNER RADIUS *****
      INCRAD = 1
      RAD = RMIN
C
C      ***** MID-LAYERS, INNER RADIUS, FIRST PHI *****
      IPHI = 1
      T1 = TOLD(INCRAD, IPHI, LAYER)
      T2 = TSTAR(INCRAD, IPHI, (LAYER+1))
      T3 = TSTAR(INCRAD, IPHI, (LAYER-1))
      T4 = TSTAR(INCRAD, IPHI, LAYER)
      T5 = T2STAR(INCRAD, NPHI, LAYER)
      T6 = T2STAR(INCRAD, (IPHI+1), LAYER)
      T7 = T2STAR(INCRAD, IPHI, LAYER)
      T8 = T3STAR((INCRAD+1), IPHI, LAYER)
      T9 = T3STAR(INCRAD, IPHI, LAYER)
      CALL BCRMI4 (TEMP)
      TNEW(INCRAD, IPHI, LAYER) = TEMP

```

```

C
C      ***** MID-LAYERS, INNER RADIUS, MID-PHIS *****
DO 600 IPHI=2, (NPHI-1)
      T1 = TOLD (INCRAD, IPHI, LAYER)
      T2 = TSTAR (INCRAD, IPHI, (LAYER+1))
      T3 = TSTAR (INCRAD, IPHI, (LAYER-1))
      T4 = TSTAR (INCRAD, IPHI, LAYER)
      T5 = T2STAR (INCRAD, (IPHI-1), LAYER)
      T6 = T2STAR (INCRAD, (IPHI+1), LAYER)
      T7 = T2STAR (INCRAD, IPHI, LAYER)
      T8 = T3STAR ((INCRAD+1), IPHI, LAYER)
      T9 = T3STAR (INCRAD, IPHI, LAYER)
      CALL BCRMI4 (TEMP)
      TNEW (INCRAD, IPHI, LAYER) = TEMP
600      CONTINUE
C
C      ***** MID-LAYERS, INNER RADIUS, LAST PHI *****
      IPHI = NPHI
      T1 = TOLD (INCRAD, IPHI, LAYER)
      T2 = TSTAR (INCRAD, IPHI, (LAYER+1))
      T3 = TSTAR (INCRAD, IPHI, (LAYER-1))
      T4 = TSTAR (INCRAD, IPHI, LAYER)
      T5 = T2STAR (INCRAD, (IPHI-1), LAYER)
      T6 = T2STAR (INCRAD, 1, LAYER)
      T7 = T2STAR (INCRAD, IPHI, LAYER)
      T8 = T3STAR ((INCRAD+1), IPHI, LAYER)
      T9 = T3STAR (INCRAD, IPHI, LAYER)
      CALL BCRMI4 (TEMP)
      TNEW (INCRAD, IPHI, LAYER) = TEMP
C
C      ***** MID-LAYERS, MID-RADII *****
DO 700 INCRAD=2, (NODES-1)
      RAD = RMIN + (INCRAD-1)*DR
C
C      ***** MID-LAYERS, MID-RADII, FIRST PHI *****
      IPHI = 1
      T1 = TOLD (INCRAD, IPHI, LAYER)
      T2 = TSTAR (INCRAD, IPHI, (LAYER+1))
      T3 = TSTAR (INCRAD, IPHI, (LAYER-1))
      T4 = TSTAR (INCRAD, IPHI, LAYER)
      T5 = T2STAR (INCRAD, NPHI, LAYER)
      T6 = T2STAR (INCRAD, (IPHI+1), LAYER)
      T7 = T2STAR (INCRAD, IPHI, LAYER)
      T8 = T3STAR ((INCRAD-1), IPHI, LAYER)
      T9 = T3STAR ((INCRAD+1), IPHI, LAYER)
      T10 = T3STAR (INCRAD, IPHI, LAYER)
      CALL BCRMM4 (TEMP)
      TNEW (INCRAD, IPHI, LAYER) = TEMP
C
C      ***** MID-LAYERS, MID-RADII, MID-PHIS *****
DO 800 IPHI=2, (NPHI-1)
      T1 = TOLD (INCRAD, IPHI, LAYER)
      T2 = TSTAR (INCRAD, IPHI, (LAYER+1))
      T3 = TSTAR (INCRAD, IPHI, (LAYER-1))
      T4 = TSTAR (INCRAD, IPHI, LAYER)

```

```

      T5 = T2STAR(INCRAD, (IPHI-1), LAYER)
      T6 = T2STAR(INCRAD, (IPHI+1), LAYER)
      T7 = T2STAR(INCRAD, IPHI, LAYER)
      T8 = T3STAR((INCRAD-1), IPHI, LAYER)
      T9 = T3STAR((INCRAD+1), IPHI, LAYER)
      T10 = T3STAR(INCRAD, IPHI, LAYER)
      CALL BCRMM4 (TEMP)
      TNEW(INCRAD, IPHI, LAYER) = TEMP
800    CONTINUE
C
C      ***** MID-LAYERS, MID-RADII, LAST PHI *****
      IPHI = NPHI
      T1 = TOLD(INCRAD, IPHI, LAYER)
      T2 = TSTAR(INCRAD, IPHI, (LAYER+1))
      T3 = TSTAR(INCRAD, IPHI, (LAYER-1))
      T4 = TSTAR(INCRAD, IPHI, LAYER)
      T5 = T2STAR(INCRAD, (IPHI-1), LAYER)
      T6 = T2STAR(INCRAD, 1, LAYER)
      T7 = T2STAR(INCRAD, IPHI, LAYER)
      T8 = T3STAR((INCRAD-1), IPHI, LAYER)
      T9 = T3STAR((INCRAD+1), IPHI, LAYER)
      T10 = T3STAR(INCRAD, IPHI, LAYER)
      CALL BCRMM4 (TEMP)
      TNEW(INCRAD, IPHI, LAYER) = TEMP
700    CONTINUE
C
C      ***** MID-LAYERS, OUTER RADII *****
      INCRAD = NODES
      RAD = RMAX
C
C      ***** MID-LAYERS, OUTER RADII, FIRST PHI *****
      IPHI = 1
      T1 = TOLD(INCRAD, IPHI, LAYER)
      T2 = TSTAR(INCRAD, IPHI, (LAYER+1))
      T3 = TSTAR(INCRAD, IPHI, (LAYER-1))
      T4 = TSTAR(INCRAD, IPHI, LAYER)
      T5 = T2STAR(INCRAD, NPHI, LAYER)
      T6 = T2STAR(INCRAD, (IPHI+1), LAYER)
      T7 = T2STAR(INCRAD, IPHI, LAYER)
      T8 = T3STAR((INCRAD-1), IPHI, LAYER)
      T9 = T3STAR(INCRAD, IPHI, LAYER)
      CALL BCRMO4 (TEMP)
      TNEW(INCRAD, IPHI, LAYER) = TEMP
C
C      ***** MID-LAYERS, OUTER RADII, MID-PHIS *****
      DO 900 IPHI=2, (NPHI-1)
      T1 = TOLD(INCRAD, IPHI, LAYER)
      T2 = TSTAR(INCRAD, IPHI, (LAYER+1))
      T3 = TSTAR(INCRAD, IPHI, (LAYER-1))
      T4 = TSTAR(INCRAD, IPHI, LAYER)
      T5 = T2STAR(INCRAD, (IPHI-1), LAYER)
      T6 = T2STAR(INCRAD, (IPHI+1), LAYER)
      T7 = T2STAR(INCRAD, IPHI, LAYER)
      T8 = T3STAR((INCRAD-1), IPHI, LAYER)
      T9 = T3STAR(INCRAD, IPHI, LAYER)

```

```

          CALL BCRMO4 (TEMP)
          TNEW(INCRAD,IPHI,LAYER) = TEMP
900      CONTINUE
C
C          ***** MID-LAYERS, OUTER RADII, LAST PHI *****
          IPHI = NPHI
          T1 = TOLD(INCRAD,IPHI,LAYER)
          T2 = TSTAR(INCRAD,IPHI,(LAYER+1))
          T3 = TSTAR(INCRAD,IPHI,(LAYER-1))
          T4 = TSTAR(INCRAD,IPHI,LAYER)
          T5 = T2STAR(INCRAD,(IPHI-1),LAYER)
          T6 = T2STAR(INCRAD,1,LAYER)
          T7 = T2STAR(INCRAD,IPHI,LAYER)
          T8 = T3STAR((INCRAD-1),IPHI,LAYER)
          T9 = T3STAR(INCRAD,IPHI,LAYER)
          CALL BCRMO4 (TEMP)
          TNEW(INCRAD,IPHI,LAYER) = TEMP
500      CONTINUE
C
CTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT TOP LAYER TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
LAYER = NLAYER
C
C          **** TOP LAYER, INNER RADIUS ****
          INCRAD = 1
          RAD = RMIN
C
C          ***** TOP LAYER, INNER RADIUS, FIRST PHI*****
          IPHI = 1
          T1 = TOLD(INCRAD,IPHI,LAYER)
          T2 = TSTAR(INCRAD,IPHI,(LAYER-1))
          T3 = TSTAR(INCRAD,IPHI,LAYER)
          T4 = T2STAR(INCRAD,NPHI,LAYER)
          T5 = T2STAR(INCRAD,(IPHI+1),LAYER)
          T6 = T2STAR(INCRAD,IPHI,LAYER)
          T7 = T3STAR((INCRAD+1),IPHI,LAYER)
          T8 = T3STAR(INCRAD,IPHI,LAYER)
          CALL BCREI4 (TEMP)
          TNEW(INCRAD,IPHI,LAYER) = TEMP
C
C          ***** TOP LAYER, INNER RADIUS, MID-PHIS *****
          DO 1000 IPHI=2,(NPHI-1)
          T1 = TOLD(INCRAD,IPHI,LAYER)
          T2 = TSTAR(INCRAD,IPHI,(LAYER-1))
          T3 = TSTAR(INCRAD,IPHI,LAYER)
          T4 = T2STAR(INCRAD,(IPHI-1),LAYER)
          T5 = T2STAR(INCRAD,(IPHI+1),LAYER)
          T6 = T2STAR(INCRAD,IPHI,LAYER)
          T7 = T3STAR((INCRAD+1),IPHI,LAYER)
          T8 = T3STAR(INCRAD,IPHI,LAYER)
          CALL BCREI4 (TEMP)
          TNEW(INCRAD,IPHI,LAYER) = TEMP
1000      CONTINUE
C
C          ***** TOP LAYER, INNER RADIUS, LAST PHI *****
          IPHI = NPHI

```

```

T1 = TOLD(INCRAD, IPHI, LAYER)
T2 = TSTAR(INCRAD, IPHI, (LAYER-1))
T3 = TSTAR(INCRAD, IPHI, LAYER)
T4 = T2STAR(INCRAD, (IPHI-1), LAYER)
T5 = T2STAR(INCRAD, 1, LAYER)
T6 = T2STAR(INCRAD, IPHI, LAYER)
T7 = T3STAR((INCRAD+1), IPHI, LAYER)
T8 = T3STAR(INCRAD, IPHI, LAYER)
CALL BCREI4 (TEMP)
TNEW(INCRAD, IPHI, LAYER) = TEMP

C
C      **** TOP LAYER, MID RADII ****
DO 1100 INCRAD=2, (NODES-1)
  RAD = RMIN + (INCRAD-1)*DR

C
C      ***** TOP LAYER, MID-RADII, FIRST PHI *****
  IPHI = 1
    T1 = TOLD(INCRAD, IPHI, LAYER)
    T2 = TSTAR(INCRAD, IPHI, (LAYER-1))
    T3 = TSTAR(INCRAD, IPHI, LAYER)
    T4 = T2STAR(INCRAD, NPHI, LAYER)
    T5 = T2STAR(INCRAD, (IPHI+1), LAYER)
    T6 = T2STAR(INCRAD, IPHI, LAYER)
    T7 = T3STAR((INCRAD-1), IPHI, LAYER)
    T8 = T3STAR((INCRAD+1), IPHI, LAYER)
    T9 = T3STAR(INCRAD, IPHI, LAYER)
    CALL BCREM4 (TEMP)
    TNEW(INCRAD, IPHI, LAYER) = TEMP

C
C      ***** TOP LAYER, MID RADII, MID-PHIS *****
DO 1200 IPHI=2, (NPHI-1)
  T1 = TOLD(INCRAD, IPHI, LAYER)
  T2 = TSTAR(INCRAD, IPHI, (LAYER-1))
  T3 = TSTAR(INCRAD, IPHI, LAYER)
  T4 = T2STAR(INCRAD, (IPHI-1), LAYER)
  T5 = T2STAR(INCRAD, (IPHI+1), LAYER)
  T6 = T2STAR(INCRAD, IPHI, LAYER)
  T7 = T3STAR((INCRAD-1), IPHI, LAYER)
  T8 = T3STAR((INCRAD+1), IPHI, LAYER)
  T9 = T3STAR(INCRAD, IPHI, LAYER)
  CALL BCREM4 (TEMP)
  TNEW(INCRAD, IPHI, LAYER) = TEMP
1200 CONTINUE

C
C      ***** TOP LAYER, MID RADII, LAST PHI *****
  IPHI = NPHI
    T1 = TOLD(INCRAD, IPHI, LAYER)
    T2 = TSTAR(INCRAD, IPHI, (LAYER-1))
    T3 = TSTAR(INCRAD, IPHI, LAYER)
    T4 = T2STAR(INCRAD, (IPHI-1), LAYER)
    T5 = T2STAR(INCRAD, 1, LAYER)
    T6 = T2STAR(INCRAD, IPHI, LAYER)
    T7 = T3STAR((INCRAD-1), IPHI, LAYER)
    T8 = T3STAR((INCRAD+1), IPHI, LAYER)
    T9 = T3STAR(INCRAD, IPHI, LAYER)

```



```

C      *      THE INNER SURFACE OF THE END LAYERS (NORTH, SOUTH)      *
C      *      OF A RIGHT-CIRCULAR CYLINDER FOR THE FIRST INTERME-      *
C      *      DIATE STEP IN Z DIRECTION OF A 3-D POLAR-CYLINDRICAL      *
C      *      COORDINATE ADAPTATION OF BRIAN'S METHOD OF FINITE      *
C      *      DIFFERENCE ANALYSIS.      *
C      *****
C      PROGRAM LINE 1767
C      SUBROUTINE BCREI1 (A,B,C,D)
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C      COMMON /CONVEC/ DKR,DKP,DKZ
C      COMMON /CONVEC/ HSUBN,HSUBS,HSUBI,HSUBO
C      COMMON /GEOMET/ DPHI,DR,DZ,RAD,RMAX,RMIN,THICK,ZMAX,LPHI
C      COMMON /HEAT/ ALFA,FO,QPSUBN,QPSUBS,QPSUBI,QPSUBO
C      COMMON /MATSIZ/ NODES,NPHI,NLAYER
C      COMMON /NPOSIT/ INCRAD,IPHI,LAYER
C      COMMON /PARAMS/ CSUBP,RHO
C      COMMON /RADN/ EPSUBN,EPSUBS,EPSUBI,EPSUBO,SIGMA
C      COMMON /TEMPS/ T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,TINF,TSUR
C      *** BODY ***
C      IF (LAYER.EQ.1) THEN
C      Q = QPSUBS
C      HE = HSUBS
C      EPSE = EPSUBS
C      A = 0.D+00
C      C = -FO
C      ELSEIF (LAYER.EQ.NLAYER) THEN
C      Q = QPSUBN
C      HE = HSUBN
C      EPSE = EPSUBN
C      A = -FO
C      C = 0.D+00
C      ENDIF
C      D1 = (DKP*DZ**2)/(2*DKZ*RAD*(RAD+DR/4)*DPHI**2)
C      D2 = (DKR*(RAD+DR/2)*DZ**2)/(DKZ*(RAD+DR/4)*DR**2)
C      D3 = DZ/DKZ
C      D4 = (RAD*DZ**2)/(DKZ*(RAD+DR/4)*DR)
C      D5 = SIGMA*(T1+TSUR)*(T1**2+TSUR**2)
C      B = 1 + FO*(1 + D3*(HE+D5*EPSE) + D4*(HSUBI+D5*EPSUBI))
C      D = T1 + FO*(D1*(T2+T3-2*T1) + D2*(T4-T1)
1      + D3*(Q+HE*TINF+D5*EPSE*TSUR)
2      + D4*(QPSUBI+HSUBI*TINF+D5*EPSUBI*TSUR)) 998 RETURN
C      END
C      C
C      C
C      C      BCREI2 FORTRAN A - (REV. 11/16/88)
C      C      *****
C      C      * BCREI2 - THIS SUBROUTINE COMPUTES THE COEFFECIENTS FOR NODES AT*
C      C      *      THE INNER SURFACE OF THE END LAYERS (NORTH, SOUTH)      *
C      C      *      OF A RIGHT-CIRCULAR CYLINDER FOR THE SECOND INTERME-      *
C      C      *      DIATE STEP (PHI DIRECTION) IN BRIAN'S METHOD OF      *
C      C      *      FINITE DIFFERENCE ANALYSIS IN 3-D POLAR-CYLINDRICAL      *
C      C      *      COORDINATES.      *

```



```

C *****
C PROGRAM LINE 1818
SUBROUTINE BCRI2 (A,B,C,D)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
COMMON /CONDC/ DKR,DKP,DKZ
COMMON /CONVEC/ HSUBN,HSUBS,HSUBI,HSUBO
COMMON /GEOMET/ DPHI,DR,DZ,RAD,RMAX,RMIN,THICK,ZMAX,LPHI
COMMON /HEAT/ ALFA,FO,QPSUBN,QPSUBS,QPSUBI,QPSUBO
COMMON /MATSIZ/ NODES,NPHI,NLAYER
COMMON /NPOSIT/ INCRAD,IPHI,LAYER
COMMON /PARAMS/ CSUBP,RHO
COMMON /RADN/ EPSUBN,EPSUBS,EPSUBI,EPSUBO,SIGMA
COMMON /TEMPS/ T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,TINF,TSUR
C *** BODY ***
IF (LAYER.EQ.1) THEN
Q = QPSUBS
HE = HSUBS
EPSE = EPSUBS
ELSEIF (LAYER.EQ.NLAYER) THEN
Q = QPSUBN
HE = HSUBN
EPSE = EPSUBN
ENDIF
D1 = (DKP*DZ**2) / (2*DKZ*RAD*(RAD+DR/4)*DPHI**2)
D2 = (DKR*(RAD+DR/2)*DZ**2) / (DKZ*(RAD+DR/4)*DR**2)
D3 = DZ/DKZ
D4 = (RAD*DZ**2) / (DKZ*(RAD+DR/4)*DR)
D5 = SIGMA*(T1+TSUR)*(T1**2+TSUR**2)
A = -D1*FO
B = 1 + FO*(2*D1 + D3*(HE+D5*EPSE) + D4*(HSUBI+D5*EPSUBI))
C = -D1*FO
C
D = T1 + FO*((T2-T3) + D2*(T4-T1) + D3*(Q-HE*TINF-D5*EPSE*TSUR)
1 + D4*(QPSUBI+HSUBI*TINF+D5*EPSUBI*TSUR))
998 RETURN
END
C
C*****
C
C BCRI3 FORTRAN A - (REV. 11/16/88)
C *****
C * BCRI3 - THIS SUBROUTINE COMPUTES THE COEFFECIENTS FOR NODES AT*
C * THE INNER SURFACE OF THE END LAYERS (NORTH, SOUTH) *
C * OF A RIGHT-CIRCULAR CYLINDER FOR THE THIRD INTERME- *
C * DIATE STEP (Z DIRECTION) IN BRIAN'S METHOD OF *
C * FINITE DIFFERENCE ANALYSIS IN 3-D POLAR-CYLINDRICAL *
C * COORDINATES. *
C *****
C PROGRAM LINE 1865
SUBROUTINE BCRI3 (A,B,C,D)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
COMMON /CONDC/ DKR,DKP,DKZ
COMMON /CONVEC/ HSUBN,HSUBS,HSUBI,HSUBO
COMMON /GEOMET/ DPHI,DR,DZ,RAD,RMAX,RMIN,THICK,ZMAX,LPHI
COMMON /HEAT/ ALFA,FO,QPSUBN,QPSUBS,QPSUBI,QPSUBO

```











```

C      *      DIFFERENCE ANALYSIS IN 3-D POLAR-CYLINDRICAL      *
C      *      COORDINATES.      *
C      ****
C      PROGRAM LINE 2216
      SUBROUTINE BCREO1 (A,B,C,D)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /CONDC/ DKR,DKP,DKZ
      COMMON /CONVEC/ HSUBN,HSUBS,HSUBI,HSUBO
      COMMON /GEOMET/ DPHI,DR,DZ,RAD,RMAX,RMIN,THICK,ZMAX,LPHI
      COMMON /HEAT/ ALFA,FO,QPSUBN,QPSUBS,QPSUBI,QPSUBO
      COMMON /MATSIZ/ NODES,NPHI,NLAYER
      COMMON /NPOSIT/ INCRAD,IPHI,LAYER
      COMMON /PARAMS/ CSUBP,RHO
      COMMON /RADN/ EPSUBN,EPSubS,EPSubI,EPSubO,SIGMA
      COMMON /TEMPS/ T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,TINF,TSUR
C      *** BODY ***

      IF (LAYER.EQ.1) THEN
      Q = QPSUBS
      EPSE = EPSubS
      HE = HSUBS
      A = 0.D+00
      C = -FO
      ELSEIF (LAYER.EQ.NLAYER) THEN
      Q = QPSUBN
      EPSE = EPSubN
      HE = HSUBN
      A = -FO
      C = 0.D+00
      ENDIF
      D1 = (DKP*DZ**2) / (2*DKZ*(RAD-DR/4)*RAD*DPHI**2)
      D2 = (DKR*(RAD-DR/2)*DZ**2) / (DKZ*(RAD-DR/4)*DR**2)
      D3 = DZ/DKZ
      D4 = (RAD*DZ**2) / (DKZ*(RAD-DR/4)*DR)
      D5 = SIGMA*(T1+TSUR)*(T1**2+TSUR**2)
      B = 1 + FO*(1 + D3*(HE+D5*EPSE) + D4*(HSUBO+D5*EPSubO))
      D = T1 + FO*(D1*(T2+T3-2*T1) + D2*(T4-T1)
      1          + D3*(Q + HE*TINF + D5*EPSE*TSUR)
      2          + D4*(QPSUBO + HSUBO*TINF + D5*EPSubO*TSUR))
998      RETURN
END
C
C      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      BCREO2 FORTRAN A - (REV. 11/16/88)
C      ****
C      * BCREO2 - THIS SUBROUTINE COMPUTES THE COEFFECIENTS FOR NODES AT*
C      *      THE OUTER SURFACE OF THE END LAYERS (NORTH, SOUTH)      *
C      *      OF A RIGHT-CIRCULAR CYLINDER FOR THE SECOND INTERMED-    *
C      *      IATE STEP (PHI DIRECTION) IN BRIAN'S METHOD OF            *
C      *      FINITE DIFFERENCE ANALYSIS IN 3-D POLAR-CYLINDRICAL      *
C      *      COORDINATES.      *
C      ****
C      PROGRAM LINE 2265
      SUBROUTINE BCREO2 (A,B,C,D)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

```

















```

C      *** NO D4, D5 OR D6 REQUIRED ***
      TNEW = T1 + FO*((T2+T3-2*T4) + D1*(T5+T6-2*T7) + D2*(T8-T10)
1          + D3*(T9-T10))
C
998   RETURN
END
C
C*****
C      BCRMO1 FORTRAN A - (REV. 11/16/88)
C      *****
C      * BCRMO1 - THIS SUBROUTINE COMPUTES THE COEFFECIENTS FOR NODES AT*
C      *          THE OUTER SURFACE OF THE MID-LAYERS                      *
C      *          OF A RIGHT-CIRCULAR CYLINDER FOR THE FIRST INTERMEDIATE*
C      *          STEP (RADIAL DIRECTION) IN BRIAN'S METHOD OF FINITE      *
C      *          DIFFERENCE ANALYSIS IN 3-D POLAR-CYLINDRICAL           *
C      *          COORDINATES.                                           *
C      *****
C      PROGRAM LINE 2604
      SUBROUTINE BCRMO1 (A,B,C,D)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /CONDUCT/ DKR,DKP,DKZ
      COMMON /CONVEC/  HSUBN,HSUBS,HSUBI,HSUBO
      COMMON /GEOMET/  DPHI,DR,DZ,RAD,RMAX,RMIN,THICK,ZMAX,LPHI
      COMMON /HEAT/    ALFA,FO,QPSUBN,QPSUBS,QPSUBI,QPSUBO
      COMMON /PARAMS/  CSUBP,RHO
      COMMON /RADN/    EPSUBN,EPSUBS,EPSUBI,EPSUBO,SIGMA
      COMMON /TEMPS/   T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,TINF,TSUR
C      ***** BODY *****
      D1 = (DKP*DZ**2)/(DKZ*RAD*(RAD-DR/4)*DPHI**2)
      D2 = (2*DKR*(RAD-DR/2)*DZ**2)/(DKZ*(RAD-DR/4)*DR**2)
      D3 = (2*RAD*DZ**2)/(DKZ*(RAD-DR/4)*DR)
C      *** NO D4 REQUIRED ***
      D5 = SIGMA*(T1+TSUR)*(T1**2+TSUR**2)
      A = -FO/2
      B = 1 + FO/2*(2 + D3*(HSUBO+D5*EPSUBO))
      C = -FO/2
      D = T1 + FO/2*(D1*(T2+T3-2*T1) + D2*(T4-T1)
1          + D3*(QPSUBO + HSUBO*TINF + D5*EPSUBO*TSUR))
998   RETURN
END
C
C*****
C      BCRMO2 FORTRAN A - (REV. 11/16/88)
C      *****
C      * BCRMO2 - THIS SUBROUTINE COMPUTES THE COEFFICIENTS FOR NODES AT*
C      *          THE OUTER SURFACE OF THE MID-LAYERS                      *
C      *          OF A RIGHT-CIRCULAR CYLINDER FOR THE SECOND INTERMED- *
C      *          IATE STEP (PHI DIRECTION) IN BRIAN'S METHOD OF          *
C      *          FINITE DIFFERENCE ANALYSIS IN 3-D POLAR-CYLINDRICAL    *
C      *          COORDINATES.                                           *
C      *****
C      PROGRAM LINE 2639
      SUBROUTINE BCRMO2 (A,B,C,D)

```







[illegible]

```

OPEN (UNIT=15,FILE='ECRDAT')
ICOUNT = 0
JCOUNT = 0
READ (14,*) NODES,NPHI,NLAYER
READ (15,*) LL,MM,NN
WRITE (*,*) 'MATRIX 1 SIZE:',NODES,NPHI,NLAYER WRITE (*,*) 'MATRIX
2 SIZE:',LL,MM,NN
WRITE (*,*)
IF ((NODES.NE.LL).OR.(NPHI.NE.MM).OR.(NLAYER.NE.NN)) THEN WRITE
(*,*) '*** ABORTED COMPARE - ARRAY SIZES OF UNEQUAL LENG 1TH ***'
GOTO 998
ENDIF
1      WRITE (11,5)
5      FORMAT (T1,'COMPARISON OF BRIAN''S METHOD TEMPERATURES
AGAINST EXP ILLICIT METHOD TEMPERATURES'//,T1,'EXPRESSED AS A
PERCENTAGE DIFFEREN 2CE'//)
WRITE (11,*) 'BCRSAT'
WRITE (11,*)
WRITE (11,*) '*** 999.0 INDICATES ZERO VALUE AS DIVISOR ***'
WRITE (11,*)
WRITE (11,*)
2      READ (14,*,END=998) BTIME,BDTIME
ICOUNT=ICOUNT+1
ITIME=BTIME
READ (15,*,END=998) ETIME,EDTIME
JCOUNT=JCOUNT+1
JTIME=ETIME
IF (ITIME.NE.JTIME) THEN
WRITE (*,*) '*** ABORTED COMPARE - TIME MISMATCH ***' WRITE (*,*)
ITIME,EDTIME
WRITE (*,*) JTIME,BDTIME
GOTO 998
ENDIF
DO 100 K=1,NLAYER
DO 200 I=1,NODES
DO 300 J=1,NPHI
READ (14,*) DMAT1
READ (15,*) DMAT2
IF (DMAT2.NE.0) THEN
PE(I,J,K) = (DMAT1-DMAT2)/DMAT2 * 100 ELSE
PE(I,J,K) = 999.
ENDIF
300      CONTINUE
200      CONTINUE
100      CONTINUE
CALL PRTARR (BTIME,BDTIME,PE,NODES,NPHI,NLAYER) GOTO 2
998     IF (ICOUNT.NE.JCOUNT) THEN
WRITE (*,*) 'COMPAR - EOF ERROR ON PROGRAM LINE 2431 OR 2433'
WRITE(11,*) 'COMPAR - EOF ERROR ON PROGRAM LINE 2431 OR 2433'
ENDIF
CLOSE (11)
CLOSE (14)
CLOSE (15)
RETURN
END

```







```

C *****
C * ERROR - SUBROUTINE TO COMPUTE ERROR FOR AND PRINT TEMPERA- *
C * TURES OF A DESIGNATED NODE FROM A VALIDATION DATA *
C * FILE. *
C *****
C PROGRAM LINE 3039
SUBROUTINE ERROR (L,M,N)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
COMMON /HEAT/ ALFA,FO,QPSUBN,QPSUBS,QPSUBI,QPSUBO
COMMON /TIMES/ DELTIM,DTMAX,FINTIM,LIMIT
DIMENSION TEMP(15,15,15)
REAL TIME
OPEN(UNIT=13,FILE='ANALYT')
OPEN(UNIT=14,FILE='BCRDAT')
OPEN(UNIT=15,FILE='BCRERV')
IF (((QPSUBO.GT.1352).AND.(QPSUBO.LT.1354)).OR.((QPSUBO.GT.14999).
1AND.(QPSUBO.LT.15001))) THEN
CONTINUE
ELSE
GOTO 998
ENDIF
C READ IN MATRIX SIZE
1 READ (14,*) NODES,NPHI,NLAYER
C WRITE HEADERS TO FILE 15 (*ERROR)
WRITE (15,*) 'BCRSAT - WITH CONDUCTION, CONVECTION AND RADIATION'
WRITE (15,*)
WRITE (15,*) 'DATA SAVED AS FILE:'
WRITE (15,*)
WRITE (15,*) 'CPU TIME:'
WRITE (15,*)
WRITE (15,*) 'GRID SIZE (RADIAL,PHI,Z DIRECTIONS):',NODES,NPHI,NLAY
1ER
WRITE (15,*)
WRITE (15,*) 'SELECTED GRID POINT :',L,M,N
WRITE (15,*)
WRITE (15,*) 'THEORETICAL MAX TIMESTEP (EXPLICIT):',DTMAX WRITE
(15,*)
WRITE (15,*) 'SELECTED TIMESTEP (SEC):',DELTIM WRITE (15,*)
WRITE (15,*) 'GRID FOURIER NUMBER (Z,DELTIM DEPENDENT): ',FO WRITE
(15,*)
WRITE (15,*) 'OUTER SURFACE HEAT FLUX (W/M**2):',QPSUBO WRITE
(15,*)
WRITE (15,*)
WRITE (15,15)
15 FORMAT (T33,'NODE',T48,'ANALYTIC',T64,'PERCENT') WRITE (15,25)
25 FORMAT (T5,'TIME',T17,'DELTIM',T30,'TEMPERATURE',T46,'TEMPERATURE'
1,T65,'ERROR'/)
C READ TIME, DELTIM AND TEMPERATURE MATRICES FROM FILE 14 (*DATA)
2 READ (14,*,END=998) TIME,DELTIM
DO 100 K=1,NLAYER
DO 200 I=1,NODES
DO 300 J=1,NPHI
READ (14,*,END=995) TEMP(I,J,K)
300 CONTINUE
200 CONTINUE

```





```

1      WRITE (17,15)
15     FORMAT (T1,'COMPARISON OF BRIANS METHOD TEMPERATURES AGAINST EXPLI
        ICIT METHOD TEMPERATURES'//,T1,'EXPRESSED AS A PERCENTAGE
        DIFFERENCE 2.'/)
        WRITE (17,*) 'BCRSAT SHORT FORMAT'
        WRITE (17,*)
        WRITE (17,*) 'SAVED AS FILE:'
        WRITE (17,*)
        WRITE (17,*) 'CPU TIME:'
        WRITE (17,*)
        WRITE (17,*) '*** 999.0 INDICATES ZERO VALUE AS DIVISOR
***' WRITE (17,*)
        WRITE (17,*)
        WRITE (17,25)
25     FORMAT (T33,'EXPLICIT',T48,'BRIANS',T64,'PERCENT') WRITE (17,35)
35     FORMAT (T5,'TIME',T30,'TEMPERATURE',T46,'TEMPERATURE',T62,'DIFFERE
1NCE'/)
2      READ (14,*,END=998) BTIME,BDTIME
        ICOUNT=ICOUNT+1
        ITIME=BTIME
        READ (15,*,END=998) ETIME,EDTIME
        JCOUNT=JCOUNT+1
        JTIME=ETIME
        IF (ITIME.NE.JTIME) THEN
            WRITE (*,*) '*** WARNING - TIME MISMATCH IN CMPSHT ***'
            WRITE (*,*) ITIME,BTIME
            WRITE (*,*) JTIME,ETIME
        ENDIF
        DO 100 K=1,NLAYER
        DO 200 I=1,NODES
        DO 300 J=1,NPHI
            READ (14,*) DMAT1
            READ (15,*) DMAT2
            IF ((I.EQ.NODES).AND.(J.EQ.2).AND.(K.EQ.NLAYER).AND.
1(DMAT2.NE.0)) THEN
                PE = (DMAT1-DMAT2)/DMAT2 * 100
                WRITE (17,45) BTIME,DMAT2,DMAT1,PE 45          FORMAT
                (T1,F9.2,T26,F14.8,T41,F14.8,T61,F9.3) ELSE
                PE = 999.
            ENDIF
300         CONTINUE
200         CONTINUE
100        CONTINUE
GOTO 2
998    IF (ICOUNT.NE.JCOUNT) THEN
        WRITE (*,*) 'CMPSHT - EOF ERROR ON PROGRAM LINE 2431 OR 2433'
        WRITE (17,*) 'CMPSHT - EOF ERROR ON PROGRAM LINE 2431 OR 2433'
    ENDIF
CLOSE (14)
CLOSE (15)
CLOSE (17)
RETURN
END

```

APPENDIX G  
APPLICATION MODEL CHARACTERISTICS

**A ASSUMPTIONS\***

**1. Satellite**

- Three-axis stabilized
- Orbit lies within and z-axis normal to the plane of the ecliptic

**2. Earth**

- Atmosphere is a diffuse scatterer of solar radiation
- Albedo coefficient is 0.30 (nominal)
- Emitted infrared heat flux is 237 W/m<sup>2</sup>
- Shadow appears cylindrical from low orbits

**3. Sun**

- Diffuse, blackbody emitter of radiation

**B OTHER PARAMETERS\***

Internal Heat Loading:	$q_{\text{int}} = 1000 \text{ W (avg)}$
Solar flux:	$q''_{\text{solar}} = 1353 \text{ W/m}^2$
Earth's IR:	$q''_{\text{IR}} = 237 \text{ W/m}^2$
Albedo coefficient:	albedo = 0.30
Temperature of deep space:	$T_{\text{sur}} = 4 \text{ K}$
Geometric shape factor: (flat plate, 320 km)	$F = 0.31$
Surface absorptivity: (white paint)	$\alpha = 0.4 \text{ (nominal)}$

Surface emissivity:  
(white paint)

$$\epsilon = 0.9$$

### Operating Limits

Non-op/turn-on

40/+40 K

Operating

-15/+40 K

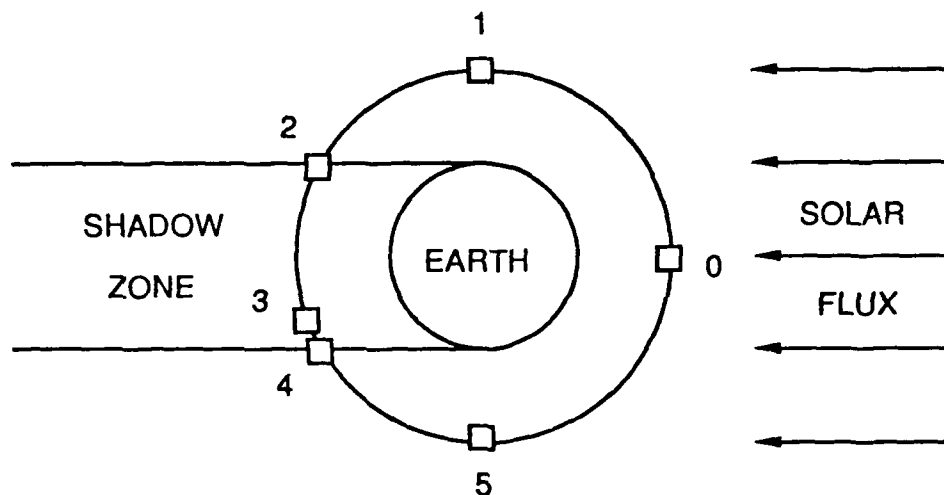


Figure 15. Orbit Time Numbering Sequence

### FLUX CONDITIONS

TIME	FLUX CONDITION
$0 \leq t' < t_1$	$q''_N = \alpha_N q''_{sun} \cos(\alpha')$ $q''_S = \alpha_s [q''_{Eir} + a q''_{sun} \cos(\alpha')]$ $q''_O = 0.5 \alpha_o q''_S \sin(\alpha')$
$t_1 \leq t' < t_2$	$q''_N = 0$ $q''_S = \alpha_s \{ q''_{Eir} + [a + \cos(\alpha')] q''_{sun} \}$ $q''_O = 0.5 \alpha_o q''_S \sin(\alpha')$
$t_2 \leq t' < t_3$	$q''_N = 0$ $q''_S = \alpha_s q''_{Eir}$

$$q''_o = 0$$

$$t_3 \leq t' < t_4$$

$$q''_N = 0$$

$$q''_s = \alpha_s \left\{ q''_{Eir} + \left| a q''_{mn} \sin (\omega x') \right| \cdot \left[ 1 - \exp \left( \frac{t_4 - t'}{\tau} \right) \right] \right\}$$

$$q''_o = 0.5 \alpha_o \left| \left[ a + q''_{mn} \sin (\omega x') \right] \cdot \left\{ 1 - \exp \frac{(t_4 - t')}{\tau} \right\} \right|$$

$$t_4 \leq t' < t_5$$

$$q''_N = 0$$

$$q''_s = \alpha_s \left\{ q''_{Eir} + \left| a q''_{mn} \sin \omega x' \right| \right\}$$

$$q''_o = 0.5 \alpha_o q''_{mn} \left| \sin (\omega x') \right|$$

where times in seconds are:

$$t_1 = 0$$

$$t_2 = 1360$$

$$t_3 = 1629$$

$$t_4 = 3757$$

$$t_5 = 3811$$

$$t_6 = 4080$$

\* Taken from Agrawal [Ref. 12]

### LIST OF REFERENCES

1. Incropera, F. P., and DeWitt, D. P., *Introduction to Heat Transfer*, John Wiley and Sons, New York, 1980.
2. Ozisik, M. N., *Heat Conduction*, John Wiley and Sons, New York, 1980.
3. Brian, P. L. T., "A Finite-Difference Method of High-Order Accuracy for the Solution of Three-Dimensional Transient Heat Conduction Problems," *American Institute of Chemical Engineers Journal*, pp. 367-370, vol. 7, no. 3, September 1961.
4. Luther, H. A., and Wilkes, J. D., *Applied Numerical Methods*, John Wiley and Sons, New York, 1969.
5. Bogert, B. P., "Some Roots of an Equation Involving Bessel Functions," *Journal of Math and Physics*, pp. 102-105, vol. 30, no. 102, 1951.
6. "IMSL Routine Name—MMBSJ0," IMSL Inc., 1978.
7. "IMSL Routine Name—MMBSJ1," IMSL Inc., 1978.
8. "IMSL Routine Name—MMBSYN," IMSL Inc., 1978.
9. "IMSL Routine Name—MMBSK0," IMSL Inc., 1978.
10. "IMSL Routine Name—MMBSK1," IMSL Inc., 1978.
11. Kaeser, D. S., *Code Optimization of Speckle Reduction Algorithms for Rocked Motor Hologram Image Processing*, M.S.M.E. Thesis, Naval Postgraduate School, Monterey, CA, December 1988.
12. Agrawal, B. N., *Design of Geosynchronous Spacecraft*, Prentice-Hall, Englewood Cliffs, 1986.

## INITIAL DISTRIBUTION LIST

	<u>No. Copies</u>
1. Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2. Mr. Duane Embree Naval Weapons Support Center Code 6042 Crane, IN 47522	2
3. Mr. Pete Wilhelm Navy Center for Space Technology, Code 8000 Naval Research Laboratory 4555 Overlook Avenue Washington, D.C. 20375-5000	2
4. Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
5. Professor R. Panholzer, Code 72 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5002	2
6. Professor A. J. Healey, Code Hy Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93943-5004	2
7. Professor Y. Joshi, Code 69Ji Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93943-5004	3
8. LCDR K. R. Heinz, USN SEAL Delivery Team ONE Naval Amphibious Base Coronado San Diego, CA 92155-5000	2

- |     |  |   |
|-----|--|---|
| 9.  | Professor A. D. Kraus, Code 62 Ks<br>Department of Electrical and Computer Engineering<br>Naval Postgraduate School<br>Monterey, CA 93943-5004 | 2 |
| 10. | Professor M. A. Melich, Code 61 Mm<br>Department of Physics<br>Naval Postgraduate School<br>Monterey, CA 93943-5004                            | 1 |
| 11. | LCDR J. A. Watson, USN<br>331 East T Street<br>Benicia, CA 94510-2256  | 1 |